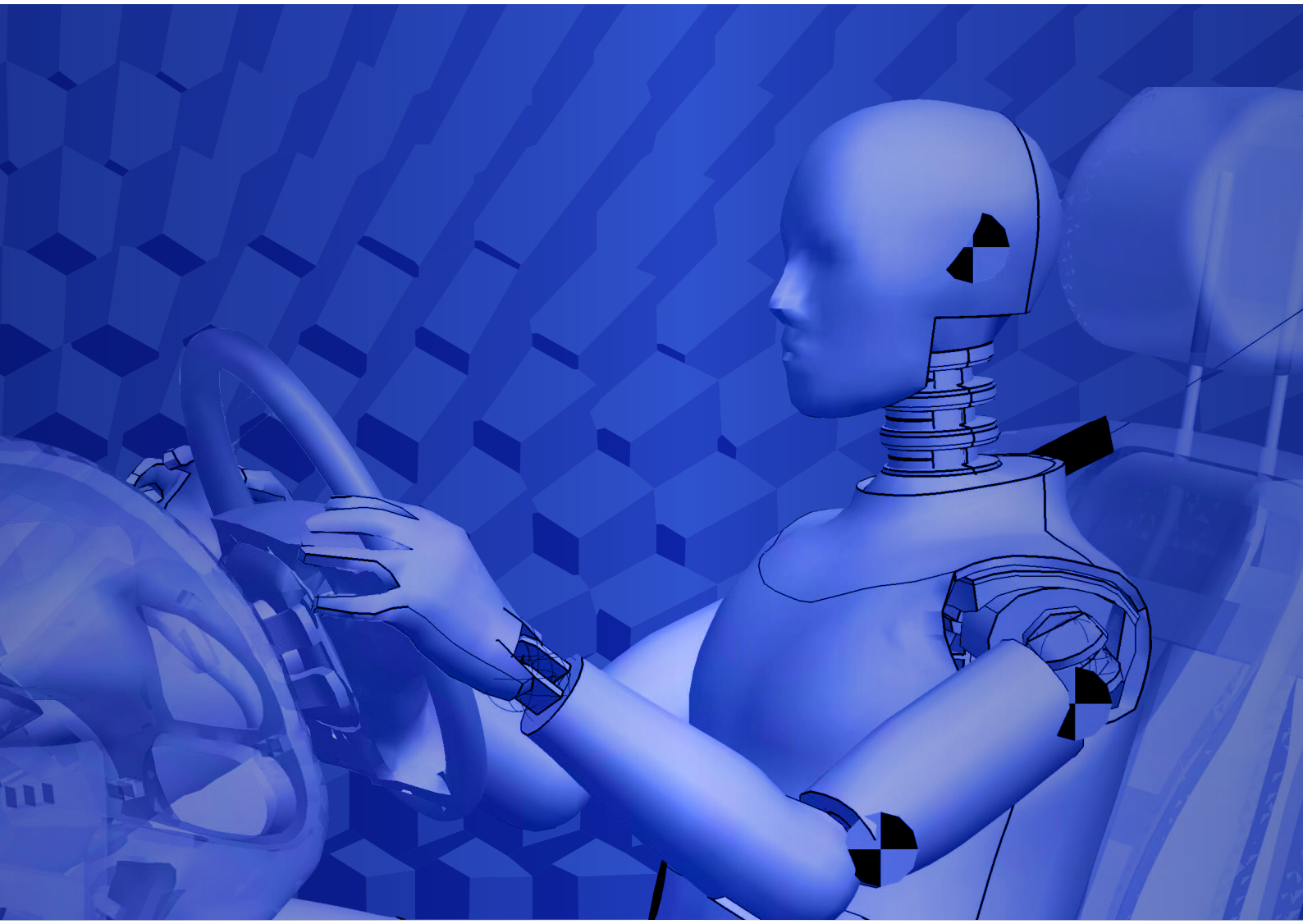
PRIMER

PRIMER JavaScript Reference Manual

from Oasys Ltd



Version 19.0

For help and support from Oasys Ltd please contact:

UK

The Arup Campus
Blythe Valley Park
Solihull
United Kingdom
B90 8AE
Tel: +44 121 213 3399
Email: dyna.support@arup.com

China

Arup China
39/F Huaihai Plaza
1045 Huaihai Road (M)
Xuhui District, Shanghai
China
200031
Tel: +86 21 3118 8875
Email: china.support@arup.com

India

Arup India Pvt Ltd
10th floor, Western Dallas Center
Plot no. 83/1, Knowledge City
Rai Durg
Hyderabad 500032
Telangana, India
Tel: +91 40 69019723 / 98
Email: india.support@arup.com

USA West

Oasys Ltd
c/o 560 Mission Street Suite 700
San Francisco
United States
CA 94105
Tel: +1 415 940 0959
Email: us.support@arup.com

Web:www.arup.com/dyna

or contact your local Oasys Ltd distributor.

LS-DYNA, LS-OPT and LS-PrePost are registered trademarks of Ansys, Inc.

global class **global 1**
Keywords **Keywords 1**
Airbag Keywords 1
Boundary Keywords 63
Comment Keywords 165
Constrained Keywords 179
Contact Keywords 456
Control Keywords 508
Damping Keywords 564
Database Keywords 565
Define Keywords 640
DeformableToRigid Keywords 831
Element Keywords 855
Hourglass Keywords 1229
Include Keywords 1244
Initial Keywords 1262
Integration Keywords 1353
Interface Keywords 1388
Load Keywords 1433
Material Keywords 1565
Node Keywords 1594
Parameter Keywords 1618
Part Keywords 1627
Rigidwall Keywords 1657
Section Keywords 1681
Sensor Keywords 1711
Set Keywords 1766
Termination Keywords 1793
Attached class **1**
Belt class **5**
Check class **33**
Colour class **39**
Conx class **41**
Dummy class **76**
File class **97**
GeometrySurface class **115**
Graphics class **135**
Group class **144**
Image class **171**
Mechanism class **174**
Model class **202**
MorphBox class **229**
MorphFlow class **253**
MorphPoint class **275**
Options class **295**
PopupWindow class **300**
Ssh class **302**
Utils class **308**
View class **313**
Widget class **320**
WidgetItem class **340**
Window class **343**
XlsxWorkbook class **359**
XlsxWorksheet class **361**
XMLParser class **364**
Xrefs class **367**
Zip class **371**

global class

The global class is the main JavaScript class. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- AllocateFlag()
- BatchMode()
- DialogueFunction(name*[function]*)
- DialogueInput(string_1, (string_2 ... string_n)*[One or more Javascript strings]*)
- DialogueInputNoEcho(string_1, (string_2 ... string_n)*[One or more Javascript strings]*)
- ErrorMessage(string*[Any valid javascript type]*)
- Execute(data*[object]*)
- Exit()
- FlagsAvailable()
- GetCurrentDirectory()
- GetInstallDirectory()
- GetPreferenceValue(program*[string]*, name*[string]*)
- GetStartInDirectory()
- Getenv(name*[string]*)
- Labels(type*[string]*, state (optional)*[boolean]*)
- MacroFunction(name*[function]*)
- Message(string*[Any valid javascript type]*)
- MilliSleep(time*[integer]*)
- NumberToString(number*[integer/real]*, width*[integer]*, pref_int (optional)*[boolean]*)
- PlayMacro(filename*[string]*, options (optional)*[Object]*)
- PlayMacro(filename*[string]*, pick (optional)*[boolean]*, view (optional)*[boolean]*, delay (optional)*[integer]*, variables (optional)*[object]*, terminate (optional)*[boolean]*) [deprecated]
- Print(string*[Any valid javascript type]*)
- Println(string*[Any valid javascript type]*)
- Requires(build*[integer]*)
- ReturnFlag(flag*[Flag]*)
- RunScript(filename*[string]*, separate (optional)*[boolean]*)
- SetCurrentDirectory(directory path*[string]*)
- SetPreferenceValue(program*[string]*, name*[string]*, value*[string]*, refresh (optional)*[boolean]*)
- Sleep(time*[integer]*)
- System(string*[Any valid javascript type]*)
- Unix()
- Use(filename*[string]*)
- UuidCreate()
- Visibility(type*[string]*, state (optional)*[boolean]*)
- WarningMessage(string*[Any valid javascript type]*)
- Windows()

Detailed Description

The global class declares the global object in JavaScript that contains the global properties and methods. As well as the core JavaScript methods, PRIMER also defines other additional ones. e.g. Message(), Print() etc. See the documentation below for more details.

Details of functions

AllocateFlag() [static]

Description

Allocate a flag for use in the script. See also ReturnFlag() and Model.PropagateFlag(). Once allocated the flag is automatically cleared for all the models currently in PRIMER.

Arguments

No arguments

Return type

Flag

Example

To allocate a flag

```
var flag = AllocateFlag();
```

BatchMode() [static]

Description

Check if PRIMER is running in batch mode (i.e. menus are not active)

Arguments

No arguments

Return type

true if in batch mode, false if not

Example

To test if PRIMER is in batch mode

```
var batch_mode = BatchMode();
```

DialogueFunction(name*[function]*) [static]

Description

Set the function for dialogue callback. This function can be used to make PRIMER return any dialogue messages that are printed. This may be useful for you to know if a particular dialogue message has been printed or a particular event has taken place.
The function will be called with 1 argument which is a string containing the dialogue message. To remove the dialogue function use DialogueFunction(null).

Arguments

Name	Type	Description
name	function	The name of the function (or null to remove the function)

Return type

No return value

Example

To set function MyDialogueFunction as the dialogue function:

```
DialogueFunction(MyDialogueFunction);
```

DialogueInput(string_1, (string_2 ... string_n)*[One or more Javascript strings]*) [static]

Description

Execute one or more lines of command line dialogue input.

Arguments

Name	Type	Description
string_1, (string_2 ... string_n)	One or more Javascript strings	The command(s) that are to be executed as if they had been typed into the dialogue box

Return type

0: No errors/warnings.
> 0: This number of errors occurred.
< 0: Absolute number is the number of warnings that occurred.

Example

To read two models:

```
DialogueInput("/re dk model_1.key 1", "/re dk model_2.key 2");
```

Note that each call to DialogueInput starts afresh at the top of the PRIMER command line "tree", so where multiple commands need to be given at sub-menu levels they need to be included in a single call. For example to restrain degrees of a mechanism assembly, and then move it by some amount:

```
DialogueInput("/mech assy " + assy_number, "fix 123", "done", "point " + point_
name, delta_x + " * *", "accept");
```

NOT:

```
DialogueInput("/mech assy " + assy_number);

DialogueInput("fix 123");
```

etc

DialogueInputNoEcho(string_1, (string_2 ... string_n)*[One or more Javascript strings]*) [static]

Description

Execute one or more lines of command line dialogue input **with no echo of commands to dialogue box**.

Arguments

Name	Type	Description
string_1, (string_2 ... string_n)	One or more Javascript strings	The command(s) that are to be executed as if they had been typed into the dialogue box

Return type

0: No errors/warnings.
> 0: This number of errors occurred.
< 0: Absolute number is the number of warnings that occurred.

Example

To read two models:

```
DialogueInputNoEcho("/re dk model_1.key 1", "/re dk model_2.key 2");
```

As with DialogueInput above each call starts at the top of the PRIMER command tree structure, so any commands destined for sub-menus must all be arguments to a single call.

ErrorMessage(string*[Any valid javascript type]*) [static]

Description

Print an error message to the dialogue box **adding a carriage return**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To print the title of model object m as an error to the dialogue box

```
ErrorMessage("The title is " + m.title);
```

Execute(data*[object]*) [static]

Description

Execute a program or script outside PRIMER and get the standard output and error streams.

Arguments

Name	Type	Description
data	object	Execute data Object has the following properties: **Name** / **Type** / **Description** arguments (optional) / Array of strings / The arguments to pass to program program / string / The program you want to run

Return type

Object with the following properties:

Name	Type	Description
status	integer	The exit code from the program/script
stderr	string	The standard error output from the program/script
stdout	string	The standard output from the program/script

Example

To run script "example.bat" with arguments "foo" and "bar":

```
var output = Execute( { program: 'example.bat', arguments: [ 'foo', 'bar' ]} );
var text   = output.stdout;
var errors = output.stderr;
var ecode  = output.status;
```

Exit() [static]

Description

Exit script

Arguments

No arguments

Return type

No return value

Example

Exit with

```
Exit();
```

FlagsAvailable() [static]

Description

Number of flags available to be used for AllocateFlag()

Arguments

No arguments

Return type

Number of flags available

Example

To get the number of flags available:

```
var flags = FlagsAvailable();
```

GetCurrentDirectory() [static]

Description

Get the current working directory

Arguments

No arguments

Return type

String containing current working directory

Example

To get the current directory:

```
var cwd = GetCurrentDirectory();
```

GetInstallDirectory() [static]

Description

Get the directory in which executables are installed. This is the OA_INSTALL environment variable, or if that is not set the directory in which the current executable is installed. Returns NULL if not found

Arguments

No arguments

Return type

string

Example

To get the install directory:

```
var install_dir = GetInstallDirectory();
```

GetPreferenceValue(program*[string]*, name*[string]*) [static]

Description

Get the Preference value with the given string in the any of admin ("OA_ADMIN") or install ("OA_INSTALL") or home ("OA_HOME") directory oa_pref

Arguments

Name	Type	Description
program	string	The program name string : Valid values are ’All’, ’D3Plot’, ’Primer’, ’Reporter’, ’Shell’, ’T/His’
name	string	The preference name string

Return type

: String containing preference value or null if preference string is not present in any oa_pref. Also if none of the above environment variables are not present, then API simply returns null. While returning preference value, locked preference value in admin and then install oa_pref takes precedence over home oa_pref. If preference is not locked in any of these oa_pref, preference in home directory oa_pref is returned.

Example

To get the preference value:

```
var pref_list = GetPreferenceValue(’All’, "font_size");
```

GetStartInDirectory() [static]

Description

Get the directory passed to PRIMER by the -start_in command line argument

Arguments

No arguments

Return type

String containing start_in directory or NULL if not set

Example

To get the start_in directory:

```
var start_in = GetStartInDirectory();
```

Getenv(name*[string]*) [static]

Description

Get the value of an environment variable

Arguments

Name	Type	Description
name	string	The environment variable name

Return type

String containing variable value or null if variable does not exist

Example

To get the value for environment variable HOME

```
var home = Getenv("HOME");
```

Labels(type*[string]*, state (optional)*[boolean]*) [static]

Description

Set or get labelling of items in PRIMER

Arguments

Name	Type	Description
type	string	The type of the item (for a list of types see Appendix I of the PRIMER manual). Additionally, to change the visibility of attached or unattached nodes you can use the types "ATTACHED_NODE" and "UNATTACHED_NODE".
state (optional)	boolean	If it is provided it is used to set the labelling status of entity. "true" to make items labelled and "false" to make them not labelled.

Return type

Boolean

Example

To turn on beam labels

```
Labels("BEAM", true);
```

To get the labelling status of beams

```
var lab = Labels("BEAM");
```

MacroFunction(name*[function]*) [static]

Description

Set the function for macro callback. This function can be used to make PRIMER return the macro command that would be recorded if macro recording was active for every button press etc. This may be useful for you to know if a particular action has been done by the user.
The function will be called with 1 argument which is a string containing the macro command. To remove the macro function use MacroFunction(null).

Arguments

Name	Type	Description
name	function	The name of the function (or null to remove a function)

Return type

No return value

Example

To set function MyMacroFunction as the macro function:

```
MacroFunction(MyMacroFunction);
```

Message(string*[Any valid javascript type]*) [static]

Description

Print a message to the dialogue box **adding a carriage return**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print. If '\r' is added to the end of the string then instead of automatically adding a carriage return in the dialogue box, the next message will overwrite the current one. This may be useful for giving feedback to the dialogue box when doing an operation.

Return type

No return value

Example

To print the title of model object m as a message to the dialogue box

```
Message("The title is " + m.title);
```

MilliSleep(time*[integer]*) [static]

Description

Pause execution of the script for *time* milliseconds. See also Sleep()

Arguments

Name	Type	Description
time	integer	Number of milliseconds to pause for

Return type

No return value

Example

To pause for 500 milliseconds

```
MilliSleep(500);
```

NumberToString(number*[integer/real]*, width*[integer]*, pref_int (optional)*[boolean]*) [static]

Description

Formats a number to a string with the specified width.

Arguments

Name	Type	Description
number	integer/real	The number you want to format.
width	integer	The width of the string you want to format it to (must be less than 80).
pref_int (optional)	boolean	By default only integer values inside the single precision 32 bit signed integer limit of approximately +/-2e9 are formatted as integers, all other numeric values are formatted as floats. With this argument set to TRUE then integer values up to the mantissa precision of a 64 bit float, approximately +/-9e15, will also be formatted as integers.

Return type

String containing the number

Example

To write the number 1.2345e+6 to a string 10 characters wide

```
var str = NumberToString(1.2345e+6, 10);
```

PlayMacro(filename*[string]*, options (optional)*[Object]*) [static]

Description

Play a macro in PRIMER

Arguments

Name	Type	Description
filename	string	The name of the macro file to play

options (optional)	Object	Options specifying how the macro file should be replayed. If omitted the default values below will be used. The properties available are: pick [logical] If picks/drags from the macro file should be replayed. If omitted the current value from macro window will be used. view [logical] If views encoded in the macro file for picks/drags should be replayed. If omitted the current value from macro window will be used. delay [integer] Delay in ms between commands when replaying. If omitted the current value from macro window will be used. variables [object] Object containing names and values for variables in the macro. If null or omitted no variables are used. terminate [logical] If the script should be terminated if an error occurs when playing the macro. If omitted the script will be terminated. utf8 [logical] If the script is UTF-8 encoded. If omitted or false the script is assumed to be ASCII text.

Return type

true if an error occured during playback, false otherwise.

Example

To play a UTF-8 encoded macro file /data/test/example.prm using the default options for picking/dragging and a delay of 500ms

```
PlayMacro("/data/test/example.prm", { delay:500, utf8:true} );
```

To play macro file /data/test/example.prm, defining values for variables A, B and C in the macro

```
PlayMacro("/data/test/example.prm", { variables: { A:10.0, B:0, C:"Example"} }
);
```

PlayMacro(filename*[string]*, pick (optional)*[boolean]*, view (optional)*[boolean]*, delay (optional)*[integer]*, variables (optional)*[object]*, terminate (optional)*[boolean]*) [static] [deprecated]

This function is deprecated in version 15.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Play a macro in PRIMER

Arguments

Name	Type	Description
filename	string	The name of the macro file to play
pick (optional)	boolean	If picks/drags from the macro file should be replayed. If omitted the current value from macro window will be used.
view (optional)	boolean	If views encoded in the macro file for picks/drags should be replayed. If omitted the current value from macro window will be used.
delay (optional)	integer	Delay in ms between commands when replaying. If omitted the current value from macro window will be used.
variables (optional)	object	Object containing names and values for variables in the macro. If null or omitted no variables are used.
terminate (optional)	boolean	If the script should be terminated if an error occurs when playing the macro. If omitted the script will be terminated.

Return type

true if an error occured during playback, false otherwise.

Example

To play macro file /data/test/example.prm using the default options for picking/dragging and the default delay

```
PlayMacro("/data/test/example.prm");
```

To play macro file /data/test/example.prm, defining values for variables A, B and C in the macro

```
var variables = new Object();
variables.A = 10.0;
variables.B = 0;
variables.C = "Example";
PlayMacro("/data/test/example.prm", true, true, 0, variables);
```

Print(string*[Any valid javascript type]*) [static]

Description

Print a string to stdout. **Note that a carriage return is not added**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To print string "Hello, world!"

```
Print("Hello, world!");
```

To print the title of model object m with a carriage return

```
print("The title is " + m.title + "\n");
```

Println(string*[Any valid javascript type]*) [static]

Description

Print a string to stdout **adding a carriage return**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To print string "Hello, world!" automatically adding a carriage return

```
Println("Hello, world!");
```

To print the title of model object m, automatically adding a carriage return

```
Println("The title is " + m.title);
```

Requires(build*[integer]*) [static]

Description

Checks to see if the build number of PRIMER is high enough to run this script. If your script requires features that are only present in builds of PRIMER greater than a certain value Require can test this and only run the script if the build is high enough.

Arguments

Name	Type	Description
build	integer	The minimum build number that is required.

Return type

No return value (if the build is not high enough the script will terminate)

Example

To only allow a script to run if the build is >= 2000

```
Requires(2000);
```

ReturnFlag(flag*[Flag]*) [static]

Description

Return a flag used in the script. See also AllocateFlag() and Model.PropagateFlag().

Arguments

Name	Type	Description
flag	Flag	The flag to return.

Return type

No return value.

Example

To return flag f:

```
ReturnFlag(f);
```

RunScript(filename*[string]*, separate (optional)*[boolean]*) [static]

Description

Run a script

Arguments

Name	Type	Description
filename	string	The name of the script file to run. If the filename is relative then the file will be searched for relative to this script. If not found then the script_directory preference will be used.
separate (optional)	boolean	If the script will use separate memory from the current script. If it uses separate memory (true) then the 'child' script is completely separated from this script and knows nothing about variables in this script. If it does not use separate memory (false) then the 'child' script will have access to all of the variables in the current script and hence variables must not clash. It is strongly recommended that you use namespaces to stop variable names from clashing. If omitted the script will use separate memory.

Return type

No return value

Example

To run script /data/test/child.js using separate memory for the child script

```
RunScript("/data/test/child.js");
```

SetCurrentDirectory(directory path*[string]*) [static]

Description

Sets the current working directory.

Arguments

Name	Type	Description
directory path	string	Path to the directory you would like to change into.

Return type

true if successful, false if not

Example

To change into the directory "/data/test" exists

```
SetCurrentDirectory("/data/test")
```

SetPreferenceValue(program*[string]*, name*[string]*, value*[string]*, refresh (optional)*[boolean]*) [static]

Description

Save the preference string and its value into oa_pref of home directory. If the preference is locked in admin ("OA_ ADMIN") or install ("OA_INSTALL") oa_pref, then API is unsuccessful. Home directory is defined by environment variable OA_HOME. If OA_HOME is not defined then API is unsuccessful.

Arguments

Name	Type	Description
program	string	The program name string : Valid values are ’All’, ’D3Plot’, ’Primer’, ’Reporter’, ’Shell’, ’T/His’
name	string	The preference name string
value	string	The preference value string. If "value" is of zero length, then the option is simply removed from the file if present, and no new entry is made.This argument cannot be null.
refresh (optional)	boolean	If the saved preference should be refreshed. If omitted, the preference will NOT be refreshed. This argument is currently only available in Primer JS API and ignored in D3PLOT and T/HIS.

Return type

An integer. Returns 0 if the preference is saved succesfully or 1 if unsuccessful

Example

To save the preference value:

```
var ierr = SetPreferenceValue( ’All’, "font_size", ’Default’);
```

Sleep(time*[integer]*) [static]

Description

Pause execution of the script for *time* seconds. See also MilliSleep()

Arguments

Name	Type	Description
time	integer	Number of seconds to pause for

Return type

No return value

Example

To pause for 2 seconds

```
Sleep(2);
```

System(string*[Any valid javascript type]*) [static]

Description

Do a system command outside PRIMER. To run an external command and get the output then please use Execute() instead.

Arguments

Name	Type	Description
string	Any valid javascript type	The system command that you want to do

Return type

integer (probably zero if command successful but is implementation-dependant)

Example

To make the directory "example"

```
System("mkdir example");
```

Unix() [static]

Description

Test whether script is running on a Unix/Linux operating system. See also Windows()

Arguments

No arguments

Return type

true if Unix/Linux, false if not

Example

To test if the OS is Unix

```
if ( Unix() )
```

Use(filename*[string]*) [static]

Description

Use script from a separate file

Arguments

Name	Type	Description
filename	string	Use allows you to include a script from a separate file. This may be useful if your script is very large and you want to split it up to help with maintenance. Alternatively you may have a 'library' of common functions which you always want to include in your scripts. Including the 'library' with Use means that any changes only have to be done in one place. Primer will look for the file in the same directory as the main script. If that fails then it will look in $OA_INSTALL/primer_library/scripts directory and the script directory specified by the *primer*script_directory* preference. **Note that the file is included when the script is compiled, NOT at runtime**.

Return type

No return value

Example

To include script from file library.js

```
Use("library.js");
```

UuidCreate() [static]

Description

Create a UUID (Universally unique ID)

Arguments

No arguments

Return type

string

Example

To create a UUID:

```
var uuid = UuidCreate();
```

Visibility(type*[string]*, state (optional)*[boolean]*) [static]

Description

Set or get visibility of items in PRIMER

Arguments

Name	Type	Description
type	string	The type of the item (for a list of types see Appendix I of the PRIMER manual). Additionally, to change the visibility of attached or unattached nodes you can use the types "ATTACHED_NODE" and "UNATTACHED_NODE".
state (optional)	boolean	If it is provided it is used to set the visibility. "true" to make items visible and "false" to make them not visible.

Return type

Boolean

Example

To make beams visible

```
Visibility("BEAM", true);
```

To get the visibility status of beams

```
var vis = Visibility("BEAM");
```

WarningMessage(string*[Any valid javascript type]*) [static]

Description

Print a warning message to the dialogue box **adding a carriage return**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to print

Return type

No return value

Example

To print the title of model object m as a warning to the dialogue box

```
WarningMessage("The title is " + m.title);
```

Windows() [static]

Description

Test whether script is running on a Windows operating system. See also Unix()

Arguments

No arguments

Return type

true if Windows, false if not

Example

To test if the OS is Windows

```
if ( Windows() )
```

Airbag class

The Airbag class gives you access to airbag cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetPropertyByIndex(index*[integer]*)
- GetPropertyByName(acronym*[string]*)
- GetPropertyByRowCol(row*[integer]*, col*[integer]*)
- GetPropertyNameForIndex(index*[integer]*)
- GetPropertyNameForRowCol(row*[integer]*, col*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- SetPropertyByIndex(index*[integer]*, value*[integer/real for numeric properties, string for character properties]*)
- SetPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*)
- SetPropertyByRowCol(row*[integer]*, col*[integer]*, value*[integer/real for numeric properties, string for character properties]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)

- Xrefs()
- toString()

Airbag constants

Name	Description
Airbag.ADIABATIC_GAS_MODEL	Airbag adiabatic gas model type
Airbag.ADVANCED_ALE	Airbag advanced ALE type
Airbag.ALE	Airbag ALE type
Airbag.HYBRID	Airbag hybrid type
Airbag.HYBRID_CHEMKIN	Airbag hybrid chemkin type
Airbag.HYBRID_JETTING	Airbag hybrid jetting type
Airbag.LINEAR_FLUID	Airbag linear fluid type
Airbag.LOAD_CURVE	Airbag load curve type
Airbag.PARTICLE	Airbag particle type
Airbag.SIMPLE_AIRBAG_MODEL	Airbag simple airbag model type
Airbag.SIMPLE_PRESSURE_VOLUME	Airbag simple pressure volume type
Airbag.WANG_NEFSKE	Airbag Wang Nefske type
Airbag.WANG_NEFSKE_JETTING	Airbag Wang Nefske jetting type
Airbag.WANG_NEFSKE_MULTIPLE_JETTING	Airbag Wang Nefske multiple jetting type

Airbag properties

Name	Type	Description
abid	integer	Airbag number. Also see the label property which is an alternative name for this.
cols	real	The number of columns of data the airbag has (read only)
exists	logical	true if airbag exists, false if referred to but not defined. (read only)
id	logical	Turns _ID on or OFF
include	integer	The Include file number that the airbag is in.
label	integer	Airbag number. Also see the abid property which is an alternative name for this.
model	integer	The Model number that the airbag is in.
properties	integer	The total number of properties that the airbag has
rows	integer	The number of rows of data the airbag has (read only). This includes the _ID card if it is set.
title	string	Airbag title
type	constant	Airbag type. Can be Airbag.SIMPLE_PRESSURE_VOLUME, Airbag.SIMPLE_AIRBAG_MODEL, Airbag.ADIABATIC_GAS_MODEL, Airbag.WANG_NEFSKE, Airbag.WANG_NEFSKE_JETTING, Airbag.WANG_NEFSKE_MULTIPLE_JETTING, Airbag.LOAD_CURVE, Airbag.LINEAR_FLUID, Airbag.HYBRID, Airbag.HYBRID_JETTING, Airbag.HYBRID_CHEMKIN, Airbag.ALE, Airbag.ADVANCED_ALE or Airbag.PARTICLE

Detailed Description

The Airbag class allows you to create, modify, edit and manipulate airbag cards. See the documentation below for more details.

Constructor

new Airbag(Model*[Model]*, type*[string]*, sid*[integer]*, sidtyp (optional)*[integer]*, abid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Airbag object.

Arguments

Name	Type	Description
Model	Model	Model that airbag will be created in
type	string	Airbag type. Can be Airbag.SIMPLE_PRESSURE_VOLUME, Airbag.SIMPLE_AIRBAG_MODEL, Airbag.ADIABATIC_GAS_MODEL, Airbag.WANG_NEFSKE, Airbag.WANG_NEFSKE_JETTING, Airbag.WANG_NEFSKE_MULTIPLE_JETTING, Airbag.LOAD_CURVE, Airbag.LINEAR_FLUID, Airbag.HYBRID, Airbag.HYBRID_JETTING, Airbag.HYBRID_CHEMKIN, Airbag.ALE, Airbag.ADVANCED_ALE or Airbag.PARTICLE
sid	integer	Set ID
sidtyp (optional)	integer	Set type: segment/part set ID
abid (optional)	integer	Airbag number
heading (optional)	string	Airbag title

Return type

Airbag object

Example

To create a new AIRBAG_SIMPLE_PRESSURE_VOLUME in model m with set ID 10 and segment set type

```
var a = new Airbag(m, Airbag.SIMPLE_PRESSURE_VOLUME, 10);
```

or

```
var a = new Airbag(m, Airbag.SIMPLE_PRESSURE_VOLUME, 10, 0);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a airbag.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the airbag

Return type

No return value

Example

To associate comment c to the airbag a:

```
a.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse airbag a:

```
a.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the airbag.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the airbag

Return type

No return value

Example

To clear flag f for airbag a:

```
a.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the airbag.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Airbag object

Example

To copy airbag a into airbag z:

```
var z = a.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an airbag.

Arguments

Name	Type	Description
Model	Model	Model that the airbag will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Airbag object (or null if not made)

Example

To start creating an airbag in model m:

```
var a = Airbag.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a airbag.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the airbag

Return type

No return value

Example

To detach comment c from the airbag a:

```
a.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit airbag a:

```
a.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for airbag. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for airbag a:

```
a.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first airbag in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first airbag in

Return type

Airbag object (or null if there are no airbags in the model).

Example

To get the first airbag in model m:

```
var a = Airbag.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free airbag label in the model. Also see Airbag.LastFreeLabel(), Airbag.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free airbag label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Airbag label.

Example

To get the first free airbag label in model m:

```
var label = Airbag.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the airbags in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all airbags will be flagged in
flag	Flag	Flag to set on the airbags

Return type

No return value

Example

To flag all of the airbags with flag f in model m:

```
Airbag.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the airbag is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the airbag

Return type

true if flagged, false if not.

Example

To check if airbag a has flag f set on it:

```
if (a.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each airbag in the model.
Note that ForEach has been designed to make looping over airbags as fast as possible and so has some limitations.
Firstly, a single temporary Airbag object is created and on each function call it is updated with the current airbag data. This means that you should not try to store the Airbag object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new airbags inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all airbags are in
func	function	Function to call for each airbag
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the airbags in model m:

```
Airbag.ForEach(m, test);
function test(a)
{
// a is Airbag object
}
```

To call function test for all of the airbags in model m with optional object:

```
var data = { x:0, y:0 };
Airbag.ForEach(m, test, data);
function test(a, extra)
{
// a is Airbag object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Airbag objects for all of the airbags in a model in Primer

Arguments

Name	Type	Description

Model	Model	Model to get airbags from

Return type

Array of Airbag objects

Example

To make an array of Airbag objects for all of the airbags in model m

```
var a = Airbag.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a airbag.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the airbag a:

```
var comm_array = a.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Airbag objects for all of the flagged airbags in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get airbags from
flag	Flag	Flag set on the airbags that you want to retrieve

Return type

Array of Airbag objects

Example

To make an array of Airbag objects for all of the airbags in model m flagged with f

```
var a = Airbag.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Airbag object for a airbag ID.

Arguments

Name	Type	Description
Model	Model	Model to find the airbag in
number	integer	number of the airbag you want the Airbag object for

Return type

Airbag object (or null if airbag does not exist).

Example

To get the Airbag object for airbag 100 in model m

```
var a = Airbag.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Airbag property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Airbag.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	airbag property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Airbag property a.example is a parameter:

```
Options.property_parameter_names = true;
if (a.GetParameter(a.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Airbag property a.example is a parameter by using the GetParameter method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

GetPropertyByIndex(index*[integer]*)

Description

Returns the value of property at index *index* for this Airbag object or null if no property exists.

Arguments

Name	Type	Description
index	integer	The index of the property value to retrieve. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for an airbag may not have adjacent indices.

Return type

Property value (real/integer)

Example

To return the property at index 3, for airbag a:

```
var prop = a.GetPropertyByIndex(3);
```

GetPropertyByName(acronym*[string]*)

Description

Returns the value of property string *acronym* for this Airbag object or null if no property exists.

Arguments

Name	Type	Description
acronym	string	The acronym of the property value to retrieve

Return type

Property value (real/integer)

Example

To return the value of HCONV for airbag a:

```
var hconv = a.GetPropertyByName("HCONV");
```

GetPropertyByRowCol(row*[integer]*, col*[integer]*)

Description

Returns the value of the property for row and col for this Airbag object or null if no property exists. **Note that columns start at 0. Rows start at 1 if the _ID option is set, at 0 otherwise**.

Arguments

Name	Type	Description
row	integer	The row of the property value to retrieve
col	integer	The column of the property value to retrieve

Return type

Property value (real/integer)

Example

To return the value of the property at row 0, column 3 for airbag a:

```
var prop = a.GetPropertyByRowCol(0, 3);
```

GetPropertyNameForIndex(index*[integer]*)

Description

Returns the name of the property at index *index* for this Airbag object or null if there is no property.

Arguments

Name	Type	Description

index	integer	The index of the property name to retrieve. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for an airbag may not have adjacent indices.

Return type

Property name (string)

Example

To return the name of the property at index 3, for airbag a:

```
var name = a.GetPropertyNameForIndex(3);
```

GetPropertyNameForRowCol(row*[integer]*, col*[integer]*)

Description

Returns the name of the property at row and col for this Airbag object or null if there is no property. **Note that columns start at 0. Rows start at 1 if the _ID option is set, at 0 otherwise**.

Arguments

Name	Type	Description
row	integer	The row of the property name to retrieve
col	integer	The column of the property name to retrieve

Return type

Property name (string)

Example

To return the name of the property at row 0, column 1 for airbag a:

```
var name = a.GetPropertyNameForRowCol(0, 1);
```

Keyword()

Description

Returns the keyword for this airbag (e.g. *AIRBAG_SIMPLE_PRESSURE_VOLUME, *AIRBAG_SIMPLE_ AIRBAG_MODEL etc). **Note that a carriage return is not added**. See also Airbag.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for airbag a:

```
var key = a.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the airbag. **Note that a carriage return is not added**. See also Airbag.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for airbag a:

```
var cards = a.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last airbag in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last airbag in

Return type

Airbag object (or null if there are no airbags in the model).

Example

To get the last airbag in model m:

```
var a = Airbag.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free airbag label in the model. Also see Airbag.FirstFreeLabel(), Airbag.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free airbag label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Airbag label.

Example

To get the last free airbag label in model m:

```
var label = Airbag.LastFreeLabel(m);
```

Next()

Description

Returns the next airbag in the model.

Arguments

No arguments

Return type

Airbag object (or null if there are no more airbags in the model).

Example

To get the airbag in model m after airbag a:

```
var a = a.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) airbag label in the model. Also see Airbag.FirstFreeLabel(), Airbag.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free airbag label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Airbag label.

Example

To get the next free airbag label in model m:

```
var label = Airbag.NextFreeLabel(m);
```

Previous()

Description

Returns the previous airbag in the model.

Arguments

No arguments

Return type

Airbag object (or null if there are no more airbags in the model).

Example

To get the airbag in model m before airbag a:

```
var a = a.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the airbags in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbags will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbags in model m, from 1000000:

```
Airbag.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged airbags in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbags will be renumbered in
flag	Flag	Flag set on the airbags that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbags in model m flagged with f, from 1000000:

```
Airbag.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select airbags using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting airbags
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only airbags from that model can be selected. If the argument is a Flag then only airbags that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any airbags can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of airbags selected or null if menu cancelled

Example

To select airbags from model m, flagging those selected with flag f, giving the prompt 'Select airbags':

```
Airbag.Select(f, 'Select airbags', m);
```

To select airbags, flagging those selected with flag f but limiting selection to airbags flagged with flag l, giving the prompt 'Select airbags':

```
Airbag.Select(f, 'Select airbags', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the airbag.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the airbag

Return type

No return value

Example

To set flag f for airbag a:

```
a.SetFlag(f);
```

SetPropertyByIndex(index*[integer]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of property at index *index* for this Airbag object

Arguments

Name	Type	Description
index	integer	The index of the property value to set. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for an airbag may not have adjacent indices.
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the property at index 3, for airbag a to be 1.234:

```
a.SetPropertyByIndex(3, 1.234);
```

SetPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of property string *acronym* for this Airbag object

Arguments

Name	Type	Description
acronym	string	The acronym of the property value to set
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the value of HCONV for airbag a to be 1.23:

```
a.SetPropertyByName("HCONV", 1.23);
```

SetPropertyByRowCol(row*[integer]*, col*[integer]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of the property for row and col for this Airbag object. **Note that columns start at 0. Rows start at 1 if the _ID option is set, at 0 otherwise**.

Arguments

Name	Type	Description
row	integer	The row of the property value to set
col	integer	The column of the property value to set
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the value of the property at row 0, column 3 for airbag a to be 0.5:

```
a.SetPropertyByRowCol(0, 3, 0.5);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the airbag. The airbag will be sketched until you either call Airbag.Unsketch(), Airbag.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag is sketched. If omitted redraw is true. If you want to sketch several airbags and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch airbag a:

```
a.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged airbags in the model. The airbags will be sketched until you either call Airbag.Unsketch(), Airbag.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbags will be sketched in
flag	Flag	Flag set on the airbags that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the airbags are sketched. If omitted redraw is true. If you want to sketch flagged airbags several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all airbags flagged with flag in model m:

```
Airbag.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of airbags in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing airbags should be counted. If false or omitted referenced but undefined airbags will also be included in the total.

Return type

number of airbags

Example

To get the total number of airbags in model m:

```
var total = Airbag.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the airbags in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all airbags will be unset in
flag	Flag	Flag to unset on the airbags

Return type

No return value

Example

To unset the flag f on all the airbags in model m:

```
Airbag.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the airbag.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag is unsketched. If omitted redraw is true. If you want to unsketch several airbags and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch airbag a:

```
a.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all airbags.

Arguments

Name	Type	Description
Model	Model	Model that all airbags will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the airbags are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbags in model m:

```
Airbag.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged airbags in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbags will be unsketched in
flag	Flag	Flag set on the airbags that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the airbags are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbags flagged with flag in model m:

```
Airbag.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Airbag object.

Example

To check if Airbag property a.example is a parameter by using the Airbag.GetParameter() method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for airbag. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for airbag a:

```
a.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this airbag.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for airbag a:

```
var xrefs = a.Xrefs();
```

---

toString()

Description

Creates a string containing the airbag data in keyword format. Note that this contains the keyword header and the keyword cards. See also Airbag.Keyword() and Airbag.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for airbag a in keyword format

```
var s = a.toString();
```

ReferenceGeometry class

The ReferenceGeometry class gives you access to define airbag reference geometry cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetNode(nid*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveNode(nid*[integer]*)
- SetFlag(flag*[Flag]*)
- SetNode(nid*[integer]*, x*[real]*, y*[real]*, z*[real]*)
- Sketch(redraw (optional)*[boolean]*)
- Spool()
- StartSpool()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ReferenceGeometry properties

Name	Type	Description
aid	integer	ReferenceGeometry number. Also see the label property which is an alternative name for this.
birth	logical	Turns _BIRTH on or off
birth_time	real	Birth time
exists	logical	true if airbag reference geometry exists, false if referred to but not defined. (read only)
id	logical	Turns _ID on or OFF
include	integer	The Include file number that the airbag reference geometry is in.
label	integer	ReferenceGeometry number. Also see the label property which is an alternative name for this.
model	integer	The Model number that the airbag reference geometry is in.
nido	integer	Node number for origin
rdt	logical	Turns _RDT on or OFF
sx	real	Scale factor in X direction
sy	real	Scale factor in Y direction
sz	real	Scale factor in Z direction

Detailed Description

The ReferenceGeometry class allows you to create, modify, edit and manipulate airbag reference geometry cards. See the documentation below for more details.

Constructor

new ReferenceGeometry(Model*[Model]*, aid (optional)*[integer]*)

Description

Create a new ReferenceGeometry object.

Arguments

Name	Type	Description
Model	Model	Model that ReferenceGeometry will be created in
aid (optional)	integer	ReferenceGeometry number to set _ID suffix

Return type

ReferenceGeometry object

Example

To create a new ReferenceGeometry in model m

```
var a = new ReferenceGeometry(m);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a airbag reference geometry.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the airbag reference geometry

Return type

No return value

Example

To associate comment c to the airbag reference geometry a:

```
a.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse airbag reference geometry a:

```
a.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the airbag reference geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the airbag reference geometry

Return type

No return value

Example

To clear flag f for airbag reference geometry a:

```
a.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the airbag reference geometry.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ReferenceGeometry object

Example

To copy airbag reference geometry a into airbag reference geometry z:

```
var z = a.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an ardt.

Arguments

Name	Type	Description
Model	Model	Model that the ardt will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ReferenceGeometry object (or null if not made)

Example

To start creating an ardt in model m:

```
var m = ReferenceGeometry.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a airbag reference geometry.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the airbag reference geometry

Return type

No return value

Example

To detach comment c from the airbag reference geometry a:

```
a.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit airbag reference geometry a:

```
a.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for airbag reference geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for airbag reference geometry a:

```
a.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first airbag reference geometry in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first airbag reference geometry in

Return type

ReferenceGeometry object (or null if there are no airbag reference geometrys in the model).

Example

To get the first airbag reference geometry in model m:

```
var a = ReferenceGeometry.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free airbag reference geometry label in the model. Also see ReferenceGeometry.LastFreeLabel(), ReferenceGeometry.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free airbag reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

ReferenceGeometry label.

Example

To get the first free airbag reference geometry label in model m:

```
var label = ReferenceGeometry.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the airbag reference geometrys in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all airbag reference geometrys will be flagged in
flag	Flag	Flag to set on the airbag reference geometrys

Return type

No return value

Example

To flag all of the airbag reference geometrys with flag f in model m:

```
ReferenceGeometry.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the airbag reference geometry is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the airbag reference geometry

Return type

true if flagged, false if not.

Example

To check if airbag reference geometry a has flag f set on it:

```
if (a.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each airbag reference geometry in the model.
Note that ForEach has been designed to make looping over airbag reference geometrys as fast as possible and so has some limitations.
Firstly, a single temporary ReferenceGeometry object is created and on each function call it is updated with the current airbag reference geometry data. This means that you should not try to store the ReferenceGeometry object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new airbag reference geometrys inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all airbag reference geometrys are in
func	function	Function to call for each airbag reference geometry
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the airbag reference geometrys in model m:

```
ReferenceGeometry.ForEach(m, test);
function test(a)
{
// a is ReferenceGeometry object
}
```

To call function test for all of the airbag reference geometrys in model m with optional object:

```
var data = { x:0, y:0 };
ReferenceGeometry.ForEach(m, test, data);
function test(a, extra)
{
// a is ReferenceGeometry object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of ReferenceGeometry objects for all of the airbag reference geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get airbag reference geometrys from

Return type

Array of ReferenceGeometry objects

Example

To make an array of ReferenceGeometry objects for all of the airbag reference geometrys in model m

```
var a = ReferenceGeometry.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a airbag reference geometry.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the airbag reference geometry a:

```
var comm_array = a.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of ReferenceGeometry objects for all of the flagged airbag reference geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get airbag reference geometrys from
flag	Flag	Flag set on the airbag reference geometrys that you want to retrieve

Return type

Array of ReferenceGeometry objects

Example

To make an array of ReferenceGeometry objects for all of the airbag reference geometrys in model m flagged with f

```
var a = ReferenceGeometry.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ReferenceGeometry object for a airbag reference geometry ID.

Arguments

Name	Type	Description
Model	Model	Model to find the airbag reference geometry in
number	integer	number of the airbag reference geometry you want the ReferenceGeometry object for

Return type

ReferenceGeometry object (or null if airbag reference geometry does not exist).

Example

To get the ReferenceGeometry object for airbag reference geometry 100 in model m

```
var a = ReferenceGeometry.GetFromID(m, 100);
```

GetNode(nid*[integer]*)

Description

Returns the reference geometry coordinates for the node

Arguments

Name	Type	Description
nid	integer	Node ID

Return type

An array containing the three reference coordinates (or null if the node is not on the reference geometry)

Example

To get the reference coordinates of node number nid on reference geometry a

```
var coords = a.GetNode(nid);
```

GetParameter(prop*[string]*)

Description

Checks if a ReferenceGeometry property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ReferenceGeometry.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	airbag reference geometry property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ReferenceGeometry property a.example is a parameter:

```
Options.property_parameter_names = true;
if (a.GetParameter(a.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ReferenceGeometry property a.example is a parameter by using the GetParameter method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this reference_geometry (*AIRBAG_REFERENCE_GEOMETRY). **Note that a carriage return is not added**. See also ReferenceGeometry.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for reference_geometry m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the reference_geometry. **Note that a carriage return is not added**. See also ReferenceGeometry.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for airbag reference geometry a:

```
var cards = b.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last airbag reference geometry in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last airbag reference geometry in

Return type

ReferenceGeometry object (or null if there are no airbag reference geometrys in the model).

Example

To get the last airbag reference geometry in model m:

```
var a = ReferenceGeometry.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free airbag reference geometry label in the model. Also see ReferenceGeometry.FirstFreeLabel(), ReferenceGeometry.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free airbag reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

ReferenceGeometry label.

Example

To get the last free airbag reference geometry label in model m:

```
var label = ReferenceGeometry.LastFreeLabel(m);
```

Next()

Description

Returns the next airbag reference geometry in the model.

Arguments

No arguments

Return type

ReferenceGeometry object (or null if there are no more airbag reference geometrys in the model).

Example

To get the airbag reference geometry in model m after airbag reference geometry a:

```
var a = a.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) airbag reference geometry label in the model. Also see ReferenceGeometry.FirstFreeLabel(), ReferenceGeometry.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free airbag reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

ReferenceGeometry label.

Example

To get the next free airbag reference geometry label in model m:

```
var label = ReferenceGeometry.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a airbag reference geometry.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only airbag reference geometrys from that model can be picked. If the argument is a Flag then only airbag reference geometrys that are flagged with *limit* can be selected. If omitted, or null, any airbag reference geometrys from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

ReferenceGeometry object (or null if not picked)

Example

To pick a airbag reference geometry from model m giving the prompt 'Pick airbag reference geometry from screen':

```
var a = ReferenceGeometry.Pick('Pick airbag reference geometry from screen', m);
```

Previous()

Description

Returns the previous airbag reference geometry in the model.

Arguments

No arguments

Return type

ReferenceGeometry object (or null if there are no more airbag reference geometrys in the model).

Example

To get the airbag reference geometry in model m before airbag reference geometry a:

```
var a = a.Previous();
```

RemoveNode(nid*[integer]*)

Description

Removes a node from the reference geometry if it is on it

Arguments

Name	Type	Description
nid	integer	Node ID

Return type

No return value.

Example

To remove node 11 from reference geometry a:

```
a.RemoveNode(11);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the airbag reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbag reference geometrys will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbag reference geometrys in model m, from 1000000:

```
ReferenceGeometry.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged airbag reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbag reference geometrys will be renumbered in
flag	Flag	Flag set on the airbag reference geometrys that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbag reference geometrys in model m flagged with f, from 1000000:

```
ReferenceGeometry.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select airbag reference geometrys using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting airbag reference geometrys
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only airbag reference geometrys from that model can be selected. If the argument is a Flag then only airbag reference geometrys that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any airbag reference geometrys can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of airbag reference geometrys selected or null if menu cancelled

Example

To select airbag reference geometrys from model m, flagging those selected with flag f, giving the prompt ’Select airbag reference geometrys’:

```
ReferenceGeometry.Select(f, ’Select airbag reference geometrys’, m);
```

To select airbag reference geometrys, flagging those selected with flag f but limiting selection to airbag reference geometrys flagged with flag l, giving the prompt ’Select airbag reference geometrys’:

```
ReferenceGeometry.Select(f, ’Select airbag reference geometrys’, l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the airbag reference geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the airbag reference geometry

Return type

No return value

Example

To set flag f for airbag reference geometry a:

```
a.SetFlag(f);
```

SetNode(nid*[integer]*, x*[real]*, y*[real]*, z*[real]*)

Description

Adds a node to the reference geometry if not already there, otherwise just changes the coordinates

Arguments

Name	Type	Description
nid	integer	Node ID
x	real	X reference coordinate
y	real	Y reference coordinate
z	real	Z reference coordinate

Return type

No return value.

Example

To add node 11 to reference geometry a with coordinates 12.0, 13.0, 14.0

```
a.SetNode(11, 12.0, 13.0, 14.0);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the airbag reference geometry. The airbag reference geometry will be sketched until you either call ReferenceGeometry.Unsketch(), ReferenceGeometry.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag reference geometry is sketched. If omitted redraw is true. If you want to sketch several airbag reference geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch airbag reference geometry a:

```
a.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged airbag reference geometrys in the model. The airbag reference geometrys will be sketched until you either call ReferenceGeometry.Unsketch(), ReferenceGeometry.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbag reference geometrys will be sketched in
flag	Flag	Flag set on the airbag reference geometrys that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the airbag reference geometrys are sketched. If omitted redraw is true. If you want to sketch flagged airbag reference geometrys several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all airbag reference geometrys flagged with flag in model m:

```
ReferenceGeometry.SketchFlagged(m, flag);
```

Spool()

Description

Spools a reference geometry, entry by entry. See also ReferenceGeometry.StartSpool

Arguments

No arguments

Return type

An array containing the node ID and the three coordinates. Returns 0 if no more items

Example

To spool reference geometry a:

```
var array;
a.StartSpool();
while ( (array = a.Spool()) )
{
    do something...
}
```

Note that the extra brackets around the assignment in the example are only required to prevent a warning when compiling the script using strict mode in the debugger. This check is to help the user find cases where they accidentally typed = but actually meant ==. Adding the extra brackets stops the check from being done.

StartSpool()

Description

Starts a reference geometry spooling operation. See also ReferenceGeometry.Spool

Arguments

No arguments

Return type

No return value

Example

To start spooling reference geometry a:

```
a.StartSpool();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of airbag reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing airbag reference geometrys should be counted. If false or omitted referenced but undefined airbag reference geometrys will also be included in the total.

Return type

number of airbag reference geometrys

Example

To get the total number of airbag reference geometrys in model m:

```
var total = ReferenceGeometry.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the airbag reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all airbag reference geometrys will be unset in
flag	Flag	Flag to unset on the airbag reference geometrys

Return type

No return value

Example

To unset the flag f on all the airbag reference geometrys in model m:

```
ReferenceGeometry.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the airbag reference geometry.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag reference geometry is unsketched. If omitted redraw is true. If you want to unsketch several airbag reference geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch airbag reference geometry a:

```
a.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all airbag reference geometrys.

Arguments

Name	Type	Description
Model	Model	Model that all airbag reference geometrys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the airbag reference geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbag reference geometrys in model m:

```
ReferenceGeometry.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged airbag reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbag reference geometrys will be unsketched in
flag	Flag	Flag set on the airbag reference geometrys that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the airbag reference geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbag reference geometrys flagged with flag in model m:

```
ReferenceGeometry.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ReferenceGeometry object.

Example

To check if ReferenceGeometry property a.example is a parameter by using the ReferenceGeometry.GetParameter() method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for airbag reference geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for airbag reference geometry a:

```
a.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this airbag reference geometry.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for airbag reference geometry a:

```
var xrefs = a.Xrefs();
```

---

toString()

Description

Creates a string containing the ReferenceGeometry data in keyword format. Note that this contains the keyword header and the keyword cards. See also ReferenceGeometry.Keyword() and ReferenceGeometry.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for ReferenceGeometry rdt in keyword format

```
var s = rdt.toString();
```

ShellReferenceGeometry class

The ShellReferenceGeometry class gives you access to airbag shell reference geometry cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetShell(eid*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveShell(eid*[integer]*)
- SetFlag(flag*[Flag]*)
- SetShell(eid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, pid (optional)*[integer]*)
- Sketch(redraw (optional)*[boolean]*)
- Spool()
- StartSpool()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ShellReferenceGeometry properties

Name	Type	Description
aid	integer	ShellReferenceGeometry number. Also see the label property which is an alternative name for this.
exists	logical	true if airbag shell reference geometry exists, false if referred to but not defined. (read only)
id	logical	Turns _ID on or OFF
include	integer	The Include file number that the airbag shell reference geometry is in.
label	integer	ShellReferenceGeometry number. Also see the label property which is an alternative name for this.
model	integer	The Model number that the airbag shell reference geometry is in.
nid	integer	Node number for origin
rdt	logical	Turns _RDT on or OFF
sx	real	Scale factor in X direction
sy	real	Scale factor in Y direction
sz	real	Scale factor in Z direction

Detailed Description

The ShellReferenceGeometry class allows you to create, modify, edit and manipulate airbag shell reference geometry cards. See the documentation below for more details.

Constructor

new ShellReferenceGeometry(Model*[Model]*, aid (optional)*[integer]*)

Description

Create a new ShellReferenceGeometry object.

Arguments

Name	Type	Description
Model	Model	Model that ShellReferenceGeometry will be created in
aid (optional)	integer	ShellReferenceGeometry number to set _ID suffix

Return type

ShellReferenceGeometry object

Example

To create a new ShellReferenceGeometry in model m

```
var a = new ShellReferenceGeometry(m);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a airbag shell reference geometry.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be attached to the airbag shell reference geometry

Return type

No return value

Example

To associate comment c to the airbag shell reference geometry a:

```
a.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse airbag shell reference geometry a:

```
a.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the airbag shell reference geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the airbag shell reference geometry

Return type

No return value

Example

To clear flag f for airbag shell reference geometry a:

```
a.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the airbag shell reference geometry.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ShellReferenceGeometry object

Example

To copy airbag shell reference geometry a into airbag shell reference geometry z:

```
var z = a.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an asrg.

Arguments

Name	Type	Description
Model	Model	Model that the asrg will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ShellReferenceGeometry object (or null if not made)

Example

To start creating an asrg in model m:

```
var m = ShellReferenceGeometry.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a airbag shell reference geometry.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the airbag shell reference geometry

Return type

No return value

Example

To detach comment c from the airbag shell reference geometry a:

```
a.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit airbag shell reference geometry a:

```
a.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for airbag shell reference geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for airbag shell reference geometry a:

```
a.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first airbag shell reference geometry in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first airbag shell reference geometry in

Return type

ShellReferenceGeometry object (or null if there are no airbag shell reference geometrys in the model).

Example

To get the first airbag shell reference geometry in model m:

```
var a = ShellReferenceGeometry.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free airbag shell reference geometry label in the model. Also see ShellReferenceGeometry.LastFreeLabel(), ShellReferenceGeometry.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free airbag shell reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

ShellReferenceGeometry label.

Example

To get the first free airbag shell reference geometry label in model m:

```
var label = ShellReferenceGeometry.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the airbag shell reference geometrys in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all airbag shell reference geometrys will be flagged in
flag	Flag	Flag to set on the airbag shell reference geometrys

Return type

No return value

Example

To flag all of the airbag shell reference geometrys with flag f in model m:

```
ShellReferenceGeometry.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the airbag shell reference geometry is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the airbag shell reference geometry

Return type

true if flagged, false if not.

Example

To check if airbag shell reference geometry a has flag f set on it:

```
if (a.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each airbag shell reference geometry in the model.
Note that ForEach has been designed to make looping over airbag shell reference geometrys as fast as possible and so has some limitations.
Firstly, a single temporary ShellReferenceGeometry object is created and on each function call it is updated with the current airbag shell reference geometry data. This means that you should not try to store the ShellReferenceGeometry object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new airbag shell reference geometrys inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all airbag shell reference geometrys are in
func	function	Function to call for each airbag shell reference geometry
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the airbag shell reference geometrys in model m:

```
ShellReferenceGeometry.ForEach(m, test);
function test(a)
{
// a is ShellReferenceGeometry object
}
```

To call function test for all of the airbag shell reference geometrys in model m with optional object:

```
var data = { x:0, y:0 };
ShellReferenceGeometry.ForEach(m, test, data);
function test(a, extra)
{
// a is ShellReferenceGeometry object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of ShellReferenceGeometry objects for all of the airbag shell reference geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get airbag shell reference geometrys from

Return type

Array of ShellReferenceGeometry objects

Example

To make an array of ShellReferenceGeometry objects for all of the airbag shell reference geometrys in model m

```
var a = ShellReferenceGeometry.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a airbag shell reference geometry.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the airbag shell reference geometry a:

```
var comm_array = a.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of ShellReferenceGeometry objects for all of the flagged airbag shell reference geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get airbag shell reference geometrys from
flag	Flag	Flag set on the airbag shell reference geometrys that you want to retrieve

Return type

Array of ShellReferenceGeometry objects

Example

To make an array of ShellReferenceGeometry objects for all of the airbag shell reference geometrys in model m flagged with f

```
var a = ShellReferenceGeometry.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ShellReferenceGeometry object for a airbag shell reference geometry ID.

Arguments

Name	Type	Description
Model	Model	Model to find the airbag shell reference geometry in
number	integer	number of the airbag shell reference geometry you want the ShellReferenceGeometry object for

Return type

ShellReferenceGeometry object (or null if airbag shell reference geometry does not exist).

Example

To get the ShellReferenceGeometry object for airbag shell reference geometry 100 in model m

```
var a = ShellReferenceGeometry.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a ShellReferenceGeometry property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ShellReferenceGeometry.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	airbag shell reference geometry property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ShellReferenceGeometry property a.example is a parameter:

```
Options.property_parameter_names = true;
if (a.GetParameter(a.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ShellReferenceGeometry property a.example is a parameter by using the GetParameter method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

GetShell(eid*[integer]*)

Description

Returns the shell reference geometry nodes and pid for the shell

Arguments

Name	Type	Description
eid	integer	Shell element ID

Return type

An array containing the four reference node labels and the part ID (or null if the shell is not on the shell reference geometry)

Example

To get the node and part data of shell number eid on shell reference geometry a

```
var data = a.GetShell(eid);
var n1 = data[0];
var n2 = data[1];
var n3 = data[2];
var n4 = data[3];
var pid = data[4];
```

Keyword()

Description

Returns the keyword for this shell_reference_geometry (*AIRBAG_SHELL_REFERENCE_GEOMETRY). **Note that a carriage return is not added**. See also ShellReferenceGeometry.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for shell_reference_geometry a:

```
var key = a.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the shell_reference_geometry. **Note that a carriage return is not added**. See also ShellReferenceGeometry.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for airbag shell reference geometry a:

```
var cards = b.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last airbag shell reference geometry in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last airbag shell reference geometry in

Return type

ShellReferenceGeometry object (or null if there are no airbag shell reference geometrys in the model).

Example

To get the last airbag shell reference geometry in model m:

```
var a = ShellReferenceGeometry.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free airbag shell reference geometry label in the model. Also see ShellReferenceGeometry.FirstFreeLabel(), ShellReferenceGeometry.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free airbag shell reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

ShellReferenceGeometry label.

Example

To get the last free airbag shell reference geometry label in model m:

```
var label = ShellReferenceGeometry.LastFreeLabel(m);
```

Next()

Description

Returns the next airbag shell reference geometry in the model.

Arguments

No arguments

Return type

ShellReferenceGeometry object (or null if there are no more airbag shell reference geometrys in the model).

Example

To get the airbag shell reference geometry in model m after airbag shell reference geometry a:

```
var a = a.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) airbag shell reference geometry label in the model. Also see ShellReferenceGeometry.FirstFreeLabel(), ShellReferenceGeometry.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free airbag shell reference geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

ShellReferenceGeometry label.

Example

To get the next free airbag shell reference geometry label in model m:

```
var label = ShellReferenceGeometry.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a airbag shell reference geometry.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only airbag shell reference geometrys from that model can be picked. If the argument is a Flag then only airbag shell reference geometrys that are flagged with *limit* can be selected. If omitted, or null, any airbag shell reference geometrys from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled ’Cancel’ which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted ’Cancel’ will be used.

Return type

ShellReferenceGeometry object (or null if not picked)

Example

To pick a airbag shell reference geometry from model m giving the prompt ’Pick airbag shell reference geometry from screen’:

```
var a = ShellReferenceGeometry.Pick(’Pick airbag shell reference geometry from screen’, m);
```

Previous()

Description

Returns the previous airbag shell reference geometry in the model.

Arguments

No arguments

Return type

ShellReferenceGeometry object (or null if there are no more airbag shell reference geometrys in the model).

Example

To get the airbag shell reference geometry in model m before airbag shell reference geometry a:

```
var a = a.Previous();
```

RemoveShell(eid*[integer]*)

Description

Removes a shell from the shell reference geometry if it is on it

Arguments

Name	Type	Description
eid	integer	Element ID

Return type

No return value.

Example

To remove shell 11 from shell reference geometry a:

```
a.RemoveShell(11);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the airbag shell reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbag shell reference geometrys will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbag shell reference geometrys in model m, from 1000000:

```
ShellReferenceGeometry.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged airbag shell reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbag shell reference geometrys will be renumbered in
flag	Flag	Flag set on the airbag shell reference geometrys that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the airbag shell reference geometrys in model m flagged with f, from 1000000:

```
ShellReferenceGeometry.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select airbag shell reference geometrys using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting airbag shell reference geometrys
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only airbag shell reference geometrys from that model can be selected. If the argument is a Flag then only airbag shell reference geometrys that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any airbag shell reference geometrys can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of airbag shell reference geometrys selected or null if menu cancelled

Example

To select airbag shell reference geometrys from model m, flagging those selected with flag f, giving the prompt 'Select airbag shell reference geometrys':

```
ShellReferenceGeometry.Select(f, 'Select airbag shell reference geometrys', m);
```

To select airbag shell reference geometrys, flagging those selected with flag f but limiting selection to airbag shell reference geometrys flagged with flag l, giving the prompt 'Select airbag shell reference geometrys':

```
ShellReferenceGeometry.Select(f, 'Select airbag shell reference geometrys', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the airbag shell reference geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the airbag shell reference geometry

Return type

No return value

Example

To set flag f for airbag shell reference geometry a:

```
a.SetFlag(f);
```

SetShell(eid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, pid (optional)*[integer]*)

Description

Adds a shell to the shell reference geometry if not already there, otherwise just changes the reference nodes

Arguments

Name	Type	Description
eid	integer	Element ID
n1	integer	Nodal point 1
n2	integer	Nodal point 2
n3	integer	Nodal point 3
n4	integer	Nodal point 4
pid (optional)	integer	Part ID (ignored by LS-DYNA). If omitted pid will be zero.

Return type

No return value.

Example

To add shell 11 to shell reference geometry a with nodal points 12, 13, 14, 15 (and part ID 0):

```
a.SetShell(11, 12, 13, 14, 15);
```

To add shell 11 to shell reference geometry a with nodal points 12, 13, 14, 15 and pid 100:

```
a.SetShell(11, 12, 13, 14, 15, 100);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the airbag shell reference geometry. The airbag shell reference geometry will be sketched until you either call ShellReferenceGeometry.Unsketch(), ShellReferenceGeometry.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag shell reference geometry is sketched. If omitted redraw is true. If you want to sketch several airbag shell reference geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch airbag shell reference geometry a:

```
a.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged airbag shell reference geometrys in the model. The airbag shell reference geometrys will be sketched until you either call ShellReferenceGeometry.Unsketch(), ShellReferenceGeometry.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged airbag shell reference geometrys will be sketched in
flag	Flag	Flag set on the airbag shell reference geometrys that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the airbag shell reference geometrys are sketched. If omitted redraw is true. If you want to sketch flagged airbag shell reference geometrys several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all airbag shell reference geometrys flagged with flag in model m:

```
ShellReferenceGeometry.SketchFlagged(m, flag);
```

Spool()

Description

Spools a shell reference geometry, entry by entry. See also ShellReferenceGeometry.StartSpool

Arguments

No arguments

Return type

Returns an array containing the shell ID and the four nodal point labels. Returns 0 if no more items

Example

To spool shell reference geometry a:

```
var array;
a.StartSpool();
while ( (array = a.Spool()) )
{
    do something...
}
```

Note that the extra brackets around the assignment in the example are only required to prevent a warning when compiling the script using strict mode in the debugger. This check is to help the user find cases where they accidentally typed = but actually meant ==. Adding the extra brackets stops the check from being done.

StartSpool()

Description

Starts a shell reference geometry spooling operation. See also ShellReferenceGeometry.Spool

Arguments

No arguments

Return type

No return value

Example

To start spooling shell reference geometry a:

```
a.StartSpool();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of airbag shell reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing airbag shell reference geometrys should be counted. If false or omitted referenced but undefined airbag shell reference geometrys will also be included in the total.

Return type

number of airbag shell reference geometrys

Example

To get the total number of airbag shell reference geometrys in model m:

```
var total = ShellReferenceGeometry.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the airbag shell reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all airbag shell reference geometrys will be unset in
flag	Flag	Flag to unset on the airbag shell reference geometrys

Return type

No return value

Example

To unset the flag f on all the airbag shell reference geometrys in model m:

```
ShellReferenceGeometry.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the airbag shell reference geometry.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the airbag shell reference geometry is unsketched. If omitted redraw is true. If you want to unsketch several airbag shell reference geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch airbag shell reference geometry a:

```
a.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all airbag shell reference geometrys.

Arguments

Name	Type	Description
Model	Model	Model that all airbag shell reference geometrys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the airbag shell reference geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbag shell reference geometrys in model m:

```
ShellReferenceGeometry.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged airbag shell reference geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all airbag shell reference geometrys will be unsketched in
flag	Flag	Flag set on the airbag shell reference geometrys that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the airbag shell reference geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all airbag shell reference geometrys flagged with flag in model m:

```
ShellReferenceGeometry.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ShellReferenceGeometry object.

Example

To check if ShellReferenceGeometry property a.example is a parameter by using the ShellReferenceGeometry.GetParameter() method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for airbag shell reference geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give

details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for airbag shell reference geometry a:

```
a.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this airbag shell reference geometry.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for airbag shell reference geometry a:

```
var xrefs = a.Xrefs();
```

---

toString()

Description

Creates a string containing the ShellReferenceGeometry data in keyword format. Note that this contains the keyword header and the keyword cards. See also ShellReferenceGeometry.Keyword() and ShellReferenceGeometry.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for ShellReferenceGeometry rdt in keyword format

```
var s = rdt.toString();
```

PrescribedAccelerometerRigid class

The PrescribedAccelerometerRigid class gives you access to define *BOUNDARY_PRESCRIBED_ACCELEROMETER_RIGID cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRow(row*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRow(row*[integer]*)
- SetFlag(flag*[Flag]*)
- SetRow(row*[integer]*, data*[Array of data]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

PrescribedAccelerometerRigid properties

Name	Type	Description
exists	logical	true if prescribed accelerometer rigid exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the prescribed accelerometer rigid is in.
model	integer	The Model number that the prescribed accelerometer rigid is in.
nrow	integer	Number of accelerometer cards. (read only)

pid	integer	Part ID for rigid body whose motion is prescribed.
solv	integer	Solver type: 1 for Gaussian elimination or 2 for linear regression.

Detailed Description

The PrescribedAccelerometerRigid class allows you to create, modify, edit and manipulate boundary prescribed accelerometer rigid cards. See the documentation below for more details.

Constructor

new PrescribedAccelerometerRigid(Model*[Model]*, pid*[integer]*, solv (optional)*[integer]*)

Description

Create a new PrescribedAccelerometerRigid object.

Arguments

Name	Type	Description
Model	Model	Model that prescribed accelerometer rigid will be created in
pid	integer	Part ID for rigid body whose motion is prescribed.
solv (optional)	integer	Solver type

Return type

PrescribedAccelerometerRigid object

Example

To create a new prescribed accelerometer rigid in model m with part ID 10 and solver type 2 (linear regression):

```
var par = new PrescribedAccelerometerRigid(m, 10, 2);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a prescribed accelerometer rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the prescribed accelerometer rigid

Return type

No return value

Example

To associate comment c to the prescribed accelerometer rigid par:

```
par.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse prescribed accelerometer rigid par:

```
par.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the prescribed accelerometer rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the prescribed accelerometer rigid

Return type

No return value

Example

To clear flag f for prescribed accelerometer rigid par:

```
par.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the prescribed accelerometer rigid.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

PrescribedAccelerometerRigid object

Example

To copy prescribed accelerometer rigid par into prescribed accelerometer rigid z:

```
var z = par.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a boundary prescribed accelerometer rigid definition.

Arguments

Name	Type	Description
Model	Model	Model that the prescribed accelerometer rigid will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

PrescribedAccelerometerRigid object (or null if not made)

Example

To start creating a boundary prescribed accelerometer rigid definition in model m:

```
var par = PrescribedAccelerometerRigid.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a prescribed accelerometer rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the prescribed accelerometer rigid

Return type

No return value

Example

To detach comment c from the prescribed accelerometer rigid par:

```
par.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit prescribed accelerometer rigid par:

```
par.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for prescribed accelerometer rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for prescribed accelerometer rigid par:

```
par.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first prescribed accelerometer rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first prescribed accelerometer rigid in

Return type

PrescribedAccelerometerRigid object (or null if there are no prescribed accelerometer rigids in the model).

Example

To get the first prescribed accelerometer rigid in model m:

```
var par = PrescribedAccelerometerRigid.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the prescribed accelerometer rigids in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed accelerometer rigids will be flagged in
flag	Flag	Flag to set on the prescribed accelerometer rigids

Return type

No return value

Example

To flag all of the prescribed accelerometer rigids with flag f in model m:

```
PrescribedAccelerometerRigid.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the prescribed accelerometer rigid is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the prescribed accelerometer rigid

Return type

true if flagged, false if not.

Example

To check if prescribed accelerometer rigid par has flag f set on it:

```
if (par.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each prescribed accelerometer rigid in the model.
Note that ForEach has been designed to make looping over prescribed accelerometer rigids as fast as possible and so has some limitations.
Firstly, a single temporary PrescribedAccelerometerRigid object is created and on each function call it is updated with the current prescribed accelerometer rigid data. This means that you should not try to store the PrescribedAccelerometerRigid object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new prescribed accelerometer rigids inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed accelerometer rigids are in
func	function	Function to call for each prescribed accelerometer rigid
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the prescribed accelerometer rigids in model m:

```
PrescribedAccelerometerRigid.ForEach(m, test);
function test(par)
{
// par is PrescribedAccelerometerRigid object
}
```

To call function test for all of the prescribed accelerometer rigids in model m with optional object:

```
var data = { x:0, y:0 };
PrescribedAccelerometerRigid.ForEach(m, test, data);
function test(par, extra)
{
// par is PrescribedAccelerometerRigid object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of PrescribedAccelerometerRigid objects for all of the prescribed accelerometer rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get prescribed accelerometer rigids from

Return type

Array of PrescribedAccelerometerRigid objects

Example

To make an array of PrescribedAccelerometerRigid objects for all of the prescribed accelerometer rigids in model m

```
var par = PrescribedAccelerometerRigid.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a prescribed accelerometer rigid.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the prescribed accelerometer rigid par:

```
var comm_array = par.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of PrescribedAccelerometerRigid objects for all of the flagged prescribed accelerometer rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get prescribed accelerometer rigids from
flag	Flag	Flag set on the prescribed accelerometer rigids that you want to retrieve

Return type

Array of PrescribedAccelerometerRigid objects

Example

To make an array of PrescribedAccelerometerRigid objects for all of the prescribed accelerometer rigids in model m flagged with f

```
var par = PrescribedAccelerometerRigid.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the PrescribedAccelerometerRigid object for a prescribed accelerometer rigid ID.

Arguments

Name	Type	Description
Model	Model	Model to find the prescribed accelerometer rigid in
number	integer	number of the prescribed accelerometer rigid you want the PrescribedAccelerometerRigid object for

Return type

PrescribedAccelerometerRigid object (or null if prescribed accelerometer rigid does not exist).

Example

To get the PrescribedAccelerometerRigid object for prescribed accelerometer rigid 100 in model m

```
var par = PrescribedAccelerometerRigid.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a PrescribedAccelerometerRigid property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the PrescribedAccelerometerRigid.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	prescribed accelerometer rigid property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if PrescribedAccelerometerRigid property par.example is a parameter:

```
Options.property_parameter_names = true;
if (par.GetParameter(par.example) ) do_something...
Options.property_parameter_names = false;
```

To check if PrescribedAccelerometerRigid property par.example is a parameter by using the GetParameter method:

```
if (par.ViewParameters().GetParameter(par.example) ) do_something...
```

GetRow(row*[integer]*)

Description

Returns the data for a row in the prescribed accelerometer rigid.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

An array of numbers containing the row variables NID, CID, LCIDX, LCIDY and LCIDZ.

Example

To get the data for the 2nd row in prescribed accelerometer rigid par:

```
var data = par.GetRow(1);
```

Keyword()

Description

Returns the keyword for this prescribed accelerometer rigid. **Note that a carriage return is not added**. See also PrescribedAccelerometerRigid.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for prescribed accelerometer rigid par:

```
var key = par.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the prescribed accelerometer rigid. **Note that a carriage return is not added**. See also PrescribedAccelerometerRigid.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for prescribed accelerometer rigid par:

```
var cards = par.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last prescribed accelerometer rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last prescribed accelerometer rigid in

Return type

PrescribedAccelerometerRigid object (or null if there are no prescribed accelerometer rigids in the model).

Example

To get the last prescribed accelerometer rigid in model m:

```
var par = PrescribedAccelerometerRigid.Last(m);
```

Next()

Description

Returns the next prescribed accelerometer rigid in the model.

Arguments

No arguments

Return type

PrescribedAccelerometerRigid object (or null if there are no more prescribed accelerometer rigids in the model).

Example

To get the prescribed accelerometer rigid in model m after prescribed accelerometer rigid par:

```
var par = par.Next();
```

Previous()

Description

Returns the previous prescribed accelerometer rigid in the model.

Arguments

No arguments

Return type

PrescribedAccelerometerRigid object (or null if there are no more prescribed accelerometer rigids in the model).

Example

To get the prescribed accelerometer rigid in model m before prescribed accelerometer rigid par:

```
var par = par.Previous();
```

RemoveRow(row*[integer]*)

Description

Removes the data for a row in *BOUNDARY_PRESCRIBED_ACCELEROMETER_RIGID.

Arguments

Name	Type	Description
row	integer	The row you want to remove the data for. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second row of data for prescribed accelerometer rigid par:

```
par.RemoveRow(1);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select prescribed accelerometer rigids using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting prescribed accelerometer rigids
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only prescribed accelerometer rigids from that model can be selected. If the argument is a Flag then only prescribed accelerometer rigids that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any prescribed accelerometer rigids can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of prescribed accelerometer rigids selected or null if menu cancelled

Example

To select prescribed accelerometer rigids from model m, flagging those selected with flag f, giving the prompt 'Select prescribed accelerometer rigids':

```
PrescribedAccelerometerRigid.Select(f, 'Select prescribed accelerometer rigids', m);
```

To select prescribed accelerometer rigids, flagging those selected with flag f but limiting selection to prescribed accelerometer rigids flagged with flag l, giving the prompt 'Select prescribed accelerometer rigids':

```
PrescribedAccelerometerRigid.Select(f, 'Select prescribed accelerometer rigids', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the prescribed accelerometer rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the prescribed accelerometer rigid

Return type

No return value

Example

To set flag f for prescribed accelerometer rigid par:

```
par.SetFlag(f);
```

SetRow(row*[integer]*, data*[Array of data]*)

Description

Sets the data for a row in *BOUNDARY_PRESCRIBED_ACCELEROMETER_RIGID.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that row indices start at 0**.
data	Array of data	The data you want to set the row to

Return type

No return value.

Example

To set the second row of data for prescribed accelerometer rigid par to be node 11, coordinate system 12, and load curves 13, 14, 15:

```
var array = [11, 12, 13, 14, 15];
par.SetRow(1, array);
```

To append a new row of data (using the same array of values):

```
par.SetRow(par.nrow, array);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of prescribed accelerometer rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing prescribed accelerometer rigids should be counted. If false or omitted referenced but undefined prescribed accelerometer rigids will also be included in the total.

Return type

number of prescribed accelerometer rigids

Example

To get the total number of prescribed accelerometer rigids in model m:

```
var total = PrescribedAccelerometerRigid.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the prescribed accelerometer rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all prescribed accelerometer rigids will be unset in
flag	Flag	Flag to unset on the prescribed accelerometer rigids

Return type

No return value

Example

To unset the flag f on all the prescribed accelerometer rigids in model m:

```
PrescribedAccelerometerRigid.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

PrescribedAccelerometerRigid object.

Example

To check if PrescribedAccelerometerRigid property par.example is a parameter by using the PrescribedAccelerometerRigid.GetParameter() method:

```
if (par.ViewParameters().GetParameter(par.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for prescribed accelerometer rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for prescribed accelerometer rigid par:

```
par.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this prescribed accelerometer rigid.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for prescribed accelerometer rigid par:

```
var xrefs = par.Xrefs();
```

---

toString()

Description

Creates a string containing the prescribed accelerometer rigid data in keyword format. Note that this contains the keyword header and the keyword cards. See also PrescribedAccelerometerRigid.Keyword() and PrescribedAccelerometerRigid.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for prescribed accelerometer rigid par in keyword format

```
var s = par.toString();
```

PrescribedOrientationRigid class

The PrescribedOrientationRigid class gives you access to define *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

PrescribedOrientationRigid constants

Name	Description
PrescribedOrientationRigid.ANGLES	Boundary is *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_ANGLES.
PrescribedOrientationRigid.DIRCOS	Boundary is *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_DIRCOS.

PrescribedOrientationRigid.EULERP	Boundary is *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_EULERP.
PrescribedOrientationRigid.VECTOR	Boundary is *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_VECTOR.

PrescribedOrientationRigid properties

Name	Type	Description
birth	real	Time prior to which the body moves freely under the action of other agents.
body	integer	Reference axes: 0 for rotations about axes fixed in PIDA or 1 for those fixed in PIDB.
death	real	Time when the body is freed from the restriction.
exists	logical	true if prescribed orientation rigid exists, false if referred to but not defined. (read only)
heading	string	PrescribedOrientationRigid heading
id	logical	true if _ID option is set, false if not.
include	integer	The Include file number that the prescribed orientation rigid is in.
intrp	integer	Interpolation method: 1 for linear interpolation or 2 for cubic spline interpolation.
intrp	integer	Interpolation method: 1 for linear interpolation or 2 for cubic spline interpolation.
iseq	integer	Specifies the sequence in which the rotations are performed.
ishft	integer	Angle shift: 1 for unaltered angle curves or 2 for angle data shift in LCIDQi curves eliminating discontinuities.
lcidc11	integer	Load curve ID specifying direction cosine C11 as function of time.
lcidc12	integer	Load curve ID specifying direction cosine C12 as function of time.
lcidc13	integer	Load curve ID specifying direction cosine C13 as function of time.
lcidc21	integer	Load curve ID specifying direction cosine C21 as function of time.
lcidc22	integer	Load curve ID specifying direction cosine C22 as function of time.
lcidc23	integer	Load curve ID specifying direction cosine C23 as function of time.
lcidc31	integer	Load curve ID specifying direction cosine C31 as function of time.
lcidc32	integer	Load curve ID specifying direction cosine C32 as function of time.
lcidc33	integer	Load curve ID specifying direction cosine C33 as function of time.
lcide1	integer	Load curve ID specifying Euler parameter e1 as function of time.
lcide2	integer	Load curve ID specifying Euler parameter e2 as function of time.
lcide3	integer	Load curve ID specifying Euler parameter e3 as function of time.
lcide4	integer	Load curve ID specifying Euler parameter e4 as function of time.
lcidq1	integer	Load curve ID specifying orientation angle q1 as function of time.
lcidq2	integer	Load curve ID specifying orientation angle q2 as function of time.
lcidq3	integer	Load curve ID specifying orientation angle q3 as function of time.
lcids	integer	Load curve ID specifying spin speed of PIDB about axis parallel to vector.
lcidv1	integer	Load curve ID specifying vector measure number v1 as function of time.
lcidv2	integer	Load curve ID specifying vector measure number v2 as function of time.
lcidv3	integer	Load curve ID specifying vector measure number v3 as function of time.
model	integer	The Model number that the prescribed orientation rigid is in.

option	constant	The Boundary Prescribed Orientation Rigid option. Can be PrescribedOrientationRigid.DIRCOS, PrescribedOrientationRigid.ANGLES, PrescribedOrientationRigid.EULERP or PrescribedOrientationRigid.VECTOR.
pida	integer	Part ID for rigid body A.
pidb	integer	Part ID for rigid body B whose orientation is prescribed.
toffset	integer	Time offset flag.
valspin	real	Constant value for spin speed of PIDB about axis parallel to vector. Used when LCIDS is 0.

Detailed Description

The PrescribedOrientationRigid class allows you to create, modify, edit and manipulate boundary prescribed orientation rigid cards. See the documentation below for more details.

Constructor

new PrescribedOrientationRigid(Model*[Model]*, option*[constant]*, pidb*[integer]*, label (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new PrescribedOrientationRigid object.

Arguments

Name	Type	Description
Model	Model	Model that prescribed orientation rigid will be created in
option	constant	Suffix for boundary prescribed orientation rigid. Can be PrescribedOrientationRigid.DIRCOS PrescribedOrientationRigid.ANGLES PrescribedOrientationRigid.EULERP PrescribedOrientationRigid.VECTOR
pidb	integer	Part ID for rigid body B whose orientation is prescribed.
label (optional)	integer	PrescribedOrientationRigid number
heading (optional)	string	Title for the PrescribedOrientationRigid

Return type

PrescribedOrientationRigid object

Example

To create a new prescribed orientation rigid in model m with part ID 10 and suffix _DIRCOS:

```
var por = new PrescribedOrientationRigid(m, PrescribedOrientationRigid.DIRCOS,
10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a prescribed orientation rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the prescribed orientation rigid

Return type

No return value

Example

To associate comment c to the prescribed orientation rigid por:

```
por.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse prescribed orientation rigid por:

```
por.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the prescribed orientation rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the prescribed orientation rigid

Return type

No return value

Example

To clear flag f for prescribed orientation rigid por:

```
por.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the prescribed orientation rigid.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

PrescribedOrientationRigid object

Example

To copy prescribed orientation rigid por into prescribed orientation rigid z:

```
var z = por.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a boundary prescribed orientation rigid definition.

Arguments

Name	Type	Description
Model	Model	Model that the prescribed orientation rigid will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

PrescribedOrientationRigid object (or null if not made)

Example

To start creating a boundary prescribed orientation rigid definition in model m:

```
var por = PrescribedOrientationRigid.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a prescribed orientation rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the prescribed orientation rigid

Return type

No return value

Example

To detach comment c from the prescribed orientation rigid por:

```
por.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit prescribed orientation rigid por:

```
por.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for prescribed orientation rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for prescribed orientation rigid por:

```
por.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first prescribed orientation rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first prescribed orientation rigid in

Return type

PrescribedOrientationRigid object (or null if there are no prescribed orientation rigids in the model).

Example

To get the first prescribed orientation rigid in model m:

```
var por = PrescribedOrientationRigid.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the prescribed orientation rigids in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed orientation rigids will be flagged in
flag	Flag	Flag to set on the prescribed orientation rigids

Return type

No return value

Example

To flag all of the prescribed orientation rigids with flag f in model m:

```
PrescribedOrientationRigid.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the prescribed orientation rigid is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the prescribed orientation rigid

Return type

true if flagged, false if not.

Example

To check if prescribed orientation rigid por has flag f set on it:

```
if (por.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each prescribed orientation rigid in the model.
Note that ForEach has been designed to make looping over prescribed orientation rigids as fast as possible and so has some limitations.
Firstly, a single temporary PrescribedOrientationRigid object is created and on each function call it is updated with the current prescribed orientation rigid data. This means that you should not try to store the PrescribedOrientationRigid object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new prescribed orientation rigids inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed orientation rigids are in
func	function	Function to call for each prescribed orientation rigid
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the prescribed orientation rigids in model m:

```
PrescribedOrientationRigid.ForEach(m, test);
function test(por)
{
// por is PrescribedOrientationRigid object
}
```

To call function test for all of the prescribed orientation rigids in model m with optional object:

```
var data = { x:0, y:0 };
PrescribedOrientationRigid.ForEach(m, test, data);
function test(por, extra)
{
// por is PrescribedOrientationRigid object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of PrescribedOrientationRigid objects for all of the prescribed orientation rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get prescribed orientation rigids from

Return type

Array of PrescribedOrientationRigid objects

Example

To make an array of PrescribedOrientationRigid objects for all of the prescribed orientation rigids in model m

```
var por = PrescribedOrientationRigid.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a prescribed orientation rigid.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the prescribed orientation rigid por:

```
var comm_array = por.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of PrescribedOrientationRigid objects for all of the flagged prescribed orientation rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get prescribed orientation rigids from
flag	Flag	Flag set on the prescribed orientation rigids that you want to retrieve

Return type

Array of PrescribedOrientationRigid objects

Example

To make an array of PrescribedOrientationRigid objects for all of the prescribed orientation rigids in model m flagged with f

```
var por = PrescribedOrientationRigid.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the PrescribedOrientationRigid object for a prescribed orientation rigid ID.

Arguments

Name	Type	Description
Model	Model	Model to find the prescribed orientation rigid in
number	integer	number of the prescribed orientation rigid you want the PrescribedOrientationRigid object for

Return type

PrescribedOrientationRigid object (or null if prescribed orientation rigid does not exist).

Example

To get the PrescribedOrientationRigid object for prescribed orientation rigid 100 in model m

```
var por = PrescribedOrientationRigid.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a PrescribedOrientationRigid property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the PrescribedOrientationRigid.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	prescribed orientation rigid property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if PrescribedOrientationRigid property por.example is a parameter:

```
Options.property_parameter_names = true;
if (por.GetParameter(por.example) ) do_something...
Options.property_parameter_names = false;
```

To check if PrescribedOrientationRigid property por.example is a parameter by using the GetParameter method:

```
if (por.ViewParameters().GetParameter(por.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this prescribed orientation rigid. **Note that a carriage return is not added**. See also PrescribedOrientationRigid.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for prescribed orientation rigid por:

```
var key = por.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the prescribed orientation rigid. **Note that a carriage return is not added**. See also PrescribedOrientationRigid.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for prescribed orientation rigid por:

```
var cards = por.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last prescribed orientation rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last prescribed orientation rigid in

Return type

PrescribedOrientationRigid object (or null if there are no prescribed orientation rigids in the model).

Example

To get the last prescribed orientation rigid in model m:

```
var por = PrescribedOrientationRigid.Last(m);
```

Next()

Description

Returns the next prescribed orientation rigid in the model.

Arguments

No arguments

Return type

PrescribedOrientationRigid object (or null if there are no more prescribed orientation rigids in the model).

Example

To get the prescribed orientation rigid in model m after prescribed orientation rigid por:

```
var por = por.Next();
```

Previous()

Description

Returns the previous prescribed orientation rigid in the model.

Arguments

No arguments

Return type

PrescribedOrientationRigid object (or null if there are no more prescribed orientation rigids in the model).

Example

To get the prescribed orientation rigid in model m before prescribed orientation rigid por:

```
var por = por.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select prescribed orientation rigids using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting prescribed orientation rigids
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only prescribed orientation rigids from that model can be selected. If the argument is a Flag then only prescribed orientation rigids that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any prescribed orientation rigids can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of prescribed orientation rigids selected or null if menu cancelled

Example

To select prescribed orientation rigids from model m, flagging those selected with flag f, giving the prompt 'Select prescribed orientation rigids':

```
PrescribedOrientationRigid.Select(f, 'Select prescribed orientation rigids', m);
```

To select prescribed orientation rigids, flagging those selected with flag f but limiting selection to prescribed orientation rigids flagged with flag l, giving the prompt 'Select prescribed orientation rigids':

```
PrescribedOrientationRigid.Select(f, 'Select prescribed orientation rigids', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the prescribed orientation rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the prescribed orientation rigid

Return type

No return value

Example

To set flag f for prescribed orientation rigid por:

```
por.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the prescribed orientation rigid. The prescribed orientation rigid will be sketched until you either call PrescribedOrientationRigid.Unsketch(), PrescribedOrientationRigid.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the prescribed orientation rigid is sketched. If omitted redraw is true. If you want to sketch several prescribed orientation rigids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch prescribed orientation rigid por:

```
por.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged prescribed orientation rigids in the model. The prescribed orientation rigids will be sketched until you either call PrescribedOrientationRigid.Unsketch(), PrescribedOrientationRigid.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged prescribed orientation rigids will be sketched in
flag	Flag	Flag set on the prescribed orientation rigids that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the prescribed orientation rigids are sketched. If omitted redraw is true. If you want to sketch flagged prescribed orientation rigids several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all prescribed orientation rigids flagged with flag in model m:

```
PrescribedOrientationRigid.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of prescribed orientation rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing prescribed orientation rigids should be counted. If false or omitted referenced but undefined prescribed orientation rigids will also be included in the total.

Return type

number of prescribed orientation rigids

Example

To get the total number of prescribed orientation rigids in model m:

```
var total = PrescribedOrientationRigid.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the prescribed orientation rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all prescribed orientation rigids will be unset in
flag	Flag	Flag to unset on the prescribed orientation rigids

Return type

No return value

Example

To unset the flag f on all the prescribed orientation rigids in model m:

```
PrescribedOrientationRigid.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the prescribed orientation rigid.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the prescribed orientation rigid is unsketched. If omitted redraw is true. If you want to unsketch several prescribed orientation rigids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch prescribed orientation rigid por:

```
por.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all prescribed orientation rigids.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed orientation rigids will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the prescribed orientation rigids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all prescribed orientation rigids in model m:

```
PrescribedOrientationRigid.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged prescribed orientation rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all prescribed orientation rigids will be unsketched in
flag	Flag	Flag set on the prescribed orientation rigids that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the prescribed orientation rigids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all prescribed orientation rigids flagged with flag in model m:

```
PrescribedOrientationRigid.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

PrescribedOrientationRigid object.

Example

To check if PrescribedOrientationRigid property por.example is a parameter by using the PrescribedOrientationRigid.GetParameter() method:

```
if (por.ViewParameters().GetParameter(por.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for prescribed orientation rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for prescribed orientation rigid por:

```
por.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this prescribed orientation rigid.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for prescribed orientation rigid por:

```
var xrefs = por.Xrefs();
```

---

toString()

Description

Creates a string containing the prescribed orientation rigid data in keyword format. Note that this contains the keyword header and the keyword cards. See also PrescribedOrientationRigid.Keyword() and PrescribedOrientationRigid.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for prescribed orientation rigid por in keyword format

```
var s = por.toString();
```

PrescribedFinalGeometry class

The PrescribedFinalGeometry class gives you access to define boundary prescribed final_geometry cards in PRIMER.
More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetData(index*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveData(index*[Integer]*)
- SetData(index*[Integer]*, nid*[integer]*, x*[real]*, y*[real]*, z*[real]*, lcid (optional)*[integer]*, death (optional)*[real]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)

- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

PrescribedFinalGeometry properties

Name	Type	Description
bpfgid	integer	PrescribedFinalGeometry number. Also see the label property which is an alternative name for this.
deathd	real	Default death time.
exists	logical	true if boundary prescribed final_geometry exists, false if referred to but not defined. (read only)
id	integer	PrescribedFinalGeometry number. Also see the label property which is an alternative name for this.
include	integer	The Include file number that the boundary prescribed final_geometry is in.
label	integer	PrescribedFinalGeometry number. Also see the bpfgid property which is an alternative name for this.
lcidf	integer	Default loadcurve number.
lines (read only)	integer	Number of lines of nodal data on the card.
model	integer	The Model number that the boundary prescribed final geometry is in.

Detailed Description

The PrescribedFinalGeometry class allows you to create, modify, edit and boundary prescribed final_geometry cards. See the documentation below for more details.

Constructor

new PrescribedFinalGeometry(Model*[Model]*, bpfgid*[PrescribedFinalGeometry]*)

Description

Create a new PrescribedFinalGeometry object.

Arguments

Name	Type	Description
Model	Model	Model that PrescribedFinalGeometry will be created in
bpfgid	PrescribedFinalGeometry	PrescribedFinalGeometry number.

Return type

PrescribedFinalGeometry object

Example

To create a new final geometry 99 in model m

```
var b = new PrescribedFinalGeometry(m, 99);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a boundary prescribed final geometry.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the boundary prescribed final geometry

Return type

No return value

Example

To associate comment c to the boundary prescribed final geometry b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the boundary prescribed final geometry

Arguments

No arguments

Return type

No return value

Example

To blank boundary prescribed final geometry b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary prescribed final geometrys in model m:

```
PrescribedFinalGeometry.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed final geometrys will be blanked in
flag	Flag	Flag set on the boundary prescribed final geometrys that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary prescribed final geometrys in model m flagged with f:

```
PrescribedFinalGeometry.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the boundary prescribed final geometry is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if boundary prescribed final geometry b is blanked:

```
if (b.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse boundary prescribed final geometry b:

```
b.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the boundary prescribed final geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the boundary prescribed final geometry

Return type

No return value

Example

To clear flag f for boundary prescribed final geometry b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the boundary prescribed final geometry.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

PrescribedFinalGeometry object

Example

To copy boundary prescribed final geometry b into boundary prescribed final geometry z:

```
var z = b.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a bpfg.

Arguments

Name	Type	Description
Model	Model	Model that the bpfg will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

PrescribedFinalGeometry object (or null if not made)

Example

To start creating a bpfg n in model m:

```
var n = PrescribedFinalGeometry.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a boundary prescribed final geometry.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the boundary prescribed final geometry

Return type

No return value

Example

To detach comment c from the boundary prescribed final geometry b:

```
b.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit boundary prescribed final geometry b:

```
b.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for boundary prescribed final geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for boundary prescribed final geometry b:

```
b.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first boundary prescribed final geometry in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first boundary prescribed final geometry in

Return type

PrescribedFinalGeometry object (or null if there are no boundary prescribed final geometrys in the model).

Example

To get the first boundary prescribed final geometry in model m:

```
var b = PrescribedFinalGeometry.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free boundary prescribed final geometry label in the model. Also see PrescribedFinalGeometry.LastFreeLabel(), PrescribedFinalGeometry.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free boundary prescribed final geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

PrescribedFinalGeometry label.

Example

To get the first free boundary prescribed final geometry label in model m:

```
var label = PrescribedFinalGeometry.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the boundary prescribed final geometrys in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be flagged in
flag	Flag	Flag to set on the boundary prescribed final geometrys

Return type

No return value

Example

To flag all of the boundary prescribed final geometrys with flag f in model m:

```
PrescribedFinalGeometry.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the boundary prescribed final geometry is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the boundary prescribed final geometry

Return type

true if flagged, false if not.

Example

To check if boundary prescribed final geometry b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each boundary prescribed final geometry in the model.
Note that ForEach has been designed to make looping over boundary prescribed final geometrys as fast as possible and so has some limitations.
Firstly, a single temporary PrescribedFinalGeometry object is created and on each function call it is updated with the current boundary prescribed final geometry data. This means that you should not try to store the PrescribedFinalGeometry object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new boundary prescribed final geometrys inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys are in
func	function	Function to call for each boundary prescribed final geometry
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the boundary prescribed final geometrys in model m:

```
PrescribedFinalGeometry.ForEach(m, test);
function test(b)
{
// b is PrescribedFinalGeometry object
}
```

To call function test for all of the boundary prescribed final geometrys in model m with optional object:

```
var data = { x:0, y:0 };
PrescribedFinalGeometry.ForEach(m, test, data);
function test(b, extra)
{
// b is PrescribedFinalGeometry object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of PrescribedFinalGeometry objects for all of the boundary prescribed final geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary prescribed final geometrys from

Return type

Array of PrescribedFinalGeometry objects

Example

To make an array of PrescribedFinalGeometry objects for all of the boundary prescribed final geometrys in model m

```
var b = PrescribedFinalGeometry.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a boundary prescribed final geometry.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the boundary prescribed final geometry b:

```
var comm_array = b.GetComments();
```

GetData(index*[integer]*)

Description

Returns data for open-ended cards for a given row number in *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY.

Arguments

Name	Type	Description
index	integer	Index of open-ended card you want the data for. **Note that indices start at 0, not 1**. 0 <= index < lines

Return type

An array containing data (NID, X, Y, Z, LCID, DEATH).

Example

To get the data for the 3rd open-ended row for boundary prescribed final geometry b:

```
var data = b.GetData(2);
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of PrescribedFinalGeometry objects for all of the flagged boundary prescribed final geometrys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary prescribed final geometrys from
flag	Flag	Flag set on the boundary prescribed final geometrys that you want to retrieve

Return type

Array of PrescribedFinalGeometry objects

Example

To make an array of PrescribedFinalGeometry objects for all of the boundary prescribed final geometrys in model m flagged with f

```
var b = PrescribedFinalGeometry.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the PrescribedFinalGeometry object for a boundary prescribed final geometry ID.

Arguments

Name	Type	Description
Model	Model	Model to find the boundary prescribed final geometry in
number	integer	number of the boundary prescribed final geometry you want the PrescribedFinalGeometry object for

Return type

PrescribedFinalGeometry object (or null if boundary prescribed final geometry does not exist).

Example

To get the PrescribedFinalGeometry object for boundary prescribed final geometry 100 in model m

```
var b = PrescribedFinalGeometry.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a PrescribedFinalGeometry property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the PrescribedFinalGeometry.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	boundary prescribed final geometry property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if PrescribedFinalGeometry property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if PrescribedFinalGeometry property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this PrescribedFinalGeometry (*BOUNDARY_PRESCRIBED_FINAL_GEOMETRY). **Note that a carriage return is not added**. See also PrescribedFinalGeometry.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for PrescribedFinalGeometry bfg:

```
var key = bfg.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the PrescribedFinalGeometry. **Note that a carriage return is not added**. See also PrescribedFinalGeometry.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for PrescribedFinalGeometry bfg:

```
var cards = bfg.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last boundary prescribed final geometry in the model.

Arguments

Name	Type	Description

Model	Model	Model to get last boundary prescribed final geometry in

Return type

PrescribedFinalGeometry object (or null if there are no boundary prescribed final geometrys in the model).

Example

To get the last boundary prescribed final geometry in model m:

```
var b = PrescribedFinalGeometry.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free boundary prescribed final geometry label in the model. Also see PrescribedFinalGeometry.FirstFreeLabel(), PrescribedFinalGeometry.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free boundary prescribed final geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

PrescribedFinalGeometry label.

Example

To get the last free boundary prescribed final geometry label in model m:

```
var label = PrescribedFinalGeometry.LastFreeLabel(m);
```

Next()

Description

Returns the next boundary prescribed final geometry in the model.

Arguments

No arguments

Return type

PrescribedFinalGeometry object (or null if there are no more boundary prescribed final geometrys in the model).

Example

To get the boundary prescribed final geometry in model m after boundary prescribed final geometry b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) boundary prescribed final geometry label in the model. Also see PrescribedFinalGeometry.FirstFreeLabel(), PrescribedFinalGeometry.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free boundary prescribed final geometry label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

PrescribedFinalGeometry label.

Example

To get the next free boundary prescribed final geometry label in model m:

```
var label = PrescribedFinalGeometry.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a boundary prescribed final geometry.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary prescribed final geometrys from that model can be picked. If the argument is a Flag then only boundary prescribed final geometrys that are flagged with *limit* can be selected. If omitted, or null, any boundary prescribed final geometrys from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

PrescribedFinalGeometry object (or null if not picked)

Example

To pick a boundary prescribed final geometry from model m giving the prompt 'Pick boundary prescribed final geometry from screen':

```
var b = PrescribedFinalGeometry.Pick('Pick boundary prescribed final geometry
from screen', m);
```

Previous()

Description

Returns the previous boundary prescribed final geometry in the model.

Arguments

No arguments

Return type

PrescribedFinalGeometry object (or null if there are no more boundary prescribed final geometrys in the model).

Example

To get the boundary prescribed final geometry in model m before boundary prescribed final geometry b:

```
var b = b.Previous();
```

RemoveData(index*[Integer]*)

Description

Removes a line of data for a *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY.

Arguments

Name	Type	Description
index	Integer	The index of the *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY data to remove. **Note that indices start at 0, not 1**. 0 <= index < lines

Return type

No return value.

Example

To remove row 2 (indices start with 0) of open-ended cards for *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY b:

```
b.RemoveData(1);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary prescribed final geometrys in model m, from 1000000:

```
PrescribedFinalGeometry.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed final geometrys will be renumbered in
flag	Flag	Flag set on the boundary prescribed final geometrys that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary prescribed final geometrys in model m flagged with f, from 1000000:

```
PrescribedFinalGeometry.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select boundary prescribed final geometrys using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting boundary prescribed final geometrys
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary prescribed final geometrys from that model can be selected. If the argument is a Flag then only boundary prescribed final geometrys that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any boundary prescribed final geometrys can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of boundary prescribed final geometrys selected or null if menu cancelled

Example

To select boundary prescribed final geometrys from model m, flagging those selected with flag f, giving the prompt 'Select boundary prescribed final geometrys':

```
PrescribedFinalGeometry.Select(f, 'Select boundary prescribed final geometrys', m);
```

To select boundary prescribed final geometrys, flagging those selected with flag f but limiting selection to boundary prescribed final geometrys flagged with flag l, giving the prompt 'Select boundary prescribed final geometrys':

```
PrescribedFinalGeometry.Select(f, 'Select boundary prescribed final geometrys', l);
```

SetData(index*[Integer]*, nid*[integer]*, x*[real]*, y*[real]*, z*[real]*, lcid (optional)*[integer]*, death (optional)*[real]*)

Description

Sets a line of data for a *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY.

Arguments

Name	Type	Description
index	Integer	The index of the *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY data to set. **Note that indices start at 0, not 1**. 0 <= index <= lines
nid	integer	Node or negative node set number.
x	real	X coordinates of final geometry.
y	real	Y coordinates of final geometry.
z	real	Z coordinates of final geometry.
lcid (optional)	integer	Loadcurve number.
death (optional)	real	Death time.

Return type

No return value.

Example

To set values for row 2 (indices start with 0) of open-ended cards for *BOUNDARY_PRESCRIBED_FINAL_GEOMETRY b with the following specification: nid, x, y, z, lcid, death are 99, 0.1, 0.2, 0.3, 88, 100.0 respectively

```
b.SetData(1, 99, 0.1, 0.2, 0.3, 88, 100.0);
```

To append a new line of data (using the same example values):

```
b.SetData(b.lines, 99, 0.1, 0.2, 0.3, 88, 100.0);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the boundary prescribed final geometry.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the boundary prescribed final geometry

Return type

No return value

Example

To set flag f for boundary prescribed final geometry b:

```
b.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the boundary prescribed final geometry. The boundary prescribed final geometry will be sketched until you either call PrescribedFinalGeometry.Unsketch(), PrescribedFinalGeometry.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed final geometry is sketched. If omitted redraw is true. If you want to sketch several boundary prescribed final geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch boundary prescribed final geometry b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged boundary prescribed final geometrys in the model. The boundary prescribed final geometrys will be sketched until you either call PrescribedFinalGeometry.Unsketch(), PrescribedFinalGeometry.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed final geometrys will be sketched in
flag	Flag	Flag set on the boundary prescribed final geometrys that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed final geometrys are sketched. If omitted redraw is true. If you want to sketch flagged boundary prescribed final geometrys several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all boundary prescribed final geometrys flagged with flag in model m:

```
PrescribedFinalGeometry.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing boundary prescribed final geometrys should be counted. If false or omitted referenced but undefined boundary prescribed final geometrys will also be included in the total.

Return type

number of boundary prescribed final geometrys

Example

To get the total number of boundary prescribed final geometrys in model m:

```
var total = PrescribedFinalGeometry.Total(m);
```

Unblank()

Description

Unblanks the boundary prescribed final geometry

Arguments

No arguments

Return type

No return value

Example

To unblank boundary prescribed final geometry b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary prescribed final geometrys in model m:

```
PrescribedFinalGeometry.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged boundary prescribed final geometrys will be unblanked in

flag	Flag	Flag set on the boundary prescribed final geometrys that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary prescribed final geometrys in model m flagged with f:

```
PrescribedFinalGeometry.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all boundary prescribed final geometrys will be unset in
flag	Flag	Flag to unset on the boundary prescribed final geometrys

Return type

No return value

Example

To unset the flag f on all the boundary prescribed final geometrys in model m:

```
PrescribedFinalGeometry.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the boundary prescribed final geometry.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed final geometry is unsketched. If omitted redraw is true. If you want to unsketch several boundary prescribed final geometrys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch boundary prescribed final geometry b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all boundary prescribed final geometrys.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed final geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boundary prescribed final geometrys in model m:

```
PrescribedFinalGeometry.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged boundary prescribed final geometrys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed final geometrys will be unsketched in
flag	Flag	Flag set on the boundary prescribed final geometrys that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed final geometrys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boundary prescribed final geometrys flagged with flag in model m:

```
PrescribedFinalGeometry.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

PrescribedFinalGeometry object.

Example

To check if PrescribedFinalGeometry property b.example is a parameter by using the PrescribedFinalGeometry.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for boundary prescribed final geometry. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for boundary prescribed final geometry b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this boundary prescribed final geometry.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for boundary prescribed final geometry b:

```
var xrefs = b.Xrefs();
```

---

toString()

Description

Creates a string containing the PrescribedFinalGeometry data in keyword format. Note that this contains the keyword header and the keyword cards. See also PrescribedFinalGeometry.Keyword() and PrescribedFinalGeometry.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for PrescribedFinalGeometry bfg in keyword format

```
var s = bfg.toString();
```

PrescribedMotion class

The PrescribedMotion class gives you access to define boundary prescribed motion cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- AnimationBackward()
- AnimationBegin(Model*[Model]*, flag*[Flag]*)
- AnimationFinish()
- AnimationForward()
- AnimationGetData()
- AnimationPause()
- AnimationPlay()
- AnimationSetData(data*[object]*)
- AnimationToEnd()
- AnimationToStart()
- AnimationToTime()
- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()

- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

PrescribedMotion constants

Name	Description
PrescribedMotion.EDGE_UVW	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_EDGE_UVW.
PrescribedMotion.FACE_XYZ	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_FACE_XYZ.
PrescribedMotion.NODE	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_NODE.
PrescribedMotion.NRBC	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_RIGID, with an NRB, not a part.
PrescribedMotion.NRBC_LOCAL	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_RIGID_LOCAL, with an NRB, not a part.
PrescribedMotion.POINT_UVW	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_POINT_UVW.
PrescribedMotion.RIGID	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_RIGID.
PrescribedMotion.RIGID_LOCAL	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_RIGID_LOCAL.
PrescribedMotion.SET	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET.
PrescribedMotion.SET_BOX	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_BOX.
PrescribedMotion.SET_EDGE_UVW	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_EDGE_UVW.
PrescribedMotion.SET_FACE_XYZ	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_FACE_XYZ.
PrescribedMotion.SET_LINE	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_LINE.
PrescribedMotion.SET_POINT_UVW	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_POINT_UVW.
PrescribedMotion.SET_SEGMENT	Prescribed motion is *BOUNDARY_PRESCRIBED_MOTION_SET_SEGMENT.

PrescribedMotion properties

Name	Type	Description
birth	real	Birth time
bndout2dynain	logical	true if _BNDOUT2DYNAIN option is set, false if not
death	real	Death time
dof	integer	Degree of freedom
exists	logical	true if boundary prescribed motion exists, false if referred to but not defined. (read only)
form	integer	Formulation type. Used for Card 6.
heading	string	PrescribedMotion heading
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the boundary prescribed motion is in.

label	integer	PrescribedMotion number.
lcid	integer	Load curve of motion vs. time
model	integer	The Model number that the boundary prescribed motion is in.
mrb	integer	Master rigid body for measuring relative displacement
nbeg	integer	Node ID of a starting node. Used for PrescribedMotion.SET_LINE
nend	integer	Node ID of a ending node. Used for PrescribedMotion.SET_LINE
node1	integer	Optional orientation node for relative displacement
node2	integer	Optional orientation node for relative displacement
offset1	real	Offset 1 for types 9-11
offset2	real	Offset 2 for types 9-11
prmr	string	String representing the name of the parameter to be output to the dynain file. Used when PrescribedMotion.bndout2dynain is set to true.
sf	real	Load curve scale factor
sfd	real	Scale factor for displacement penalty stiffness. Used for Card 6.
sfr	real	Scale factor for rotational penalty stiffness. Used for Card 6.
type	constant	The Prescribed motion type. Can be PrescribedMotion.NODE, PrescribedMotion.SET, PrescribedMotion.RIGID, PrescribedMotion.RIGID_LOCAL, PrescribedMotion.NRBC, PrescribedMotion.NRBC_LOCAL, PrescribedMotion.SET_BOX, PrescribedMotion.SET_SEGMENT, PrescribedMotion.SET_LINE, PrescribedMotion.POINT_UVW, PrescribedMotion.EDGE_UVW, PrescribedMotion.FACE_XYZ, PrescribedMotion.SET_POINT_UVW, PrescribedMotion.SET_EDGE_UVW or PrescribedMotion.SET_FACE_XYZ
typeid	integer	Node ID, node set ID, part ID or NRB
vad	integer	Velocity/acceleration/displacement flag
vid	integer	Vector ID

Detailed Description

The PrescribedMotion class allows you to create, modify, edit and boundary prescribed motion cards. See the documentation below for more details.

Constructor

new PrescribedMotion(Model*[Model]*, typeid*[integer]*, dof*[integer]*, vad*[integer]*, lcid*[integer]*, type*[constant]*, label (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new PrescribedMotion object.

Arguments

Name	Type	Description
Model	Model	Model that PrescribedMotion will be created in
typeid	integer	Node ID, node set ID or part ID
dof	integer	Degree of freedom
vad	integer	Velocity/acceleration/displacement flag

lcid	integer	Load curve for motion
type	constant	Specify the type of prescribed motion (Can be PrescribedMotion.NODE, PrescribedMotion.SET, PrescribedMotion.RIGID, PrescribedMotion.RIGID_LOCAL, PrescribedMotion.NRBC, PrescribedMotion.NRBC_LOCAL, PrescribedMotion.SET_BOX, PrescribedMotion.SET_SEGMENT, PrescribedMotion.SET_LINE, PrescribedMotion.POINT_UVW, PrescribedMotion.EDGE_UVW, PrescribedMotion.FACE_XYZ, PrescribedMotion.SET_POINT_UVW, PrescribedMotion.SET_EDGE_UVW or PrescribedMotion.SET_FACE_XYZ)
label (optional)	integer	PrescribedMotion number
heading (optional)	string	Title for the PrescribedMotion

Return type

PrescribedMotion object

Example

To create a new displacement for node 100 in x using loadcurve 10 model m with label 200, of type SET

```
var b = new PrescribedMotion(m, 100, 1, 2, 10, PrescribedMotion.SET, 200);
```

Details of functions

AnimationBackward() [static]

Description

Moves backward one frame of a PrescribedMotion animation (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To move backward one frame of an animation:

```
PrescribedMotion.AnimationBackward();
```

AnimationBegin(Model*[Model]*, flag*[Flag]*) [static]

Description

Begins a PrescribedMotion animation. This **MUST** be called before any of the other Animation methods. Also see the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

Name	Type	Description
Model	Model	Model that PrescribedMotions are in
flag	Flag	Flag set on the PrescribedMotions that you want to animate

Return type

Object with the following properties:

Name	Type	Description
end	real	End time
frame	integer	Current frame
frames	integer	Number of frames
rate	integer	Animation speed in fps
repeat	integer	Animation repeat (0=off, 1=on)
start	real	Start time
time	real	Current time
timestep	real	Timestep

Example

To begin an animation of the PrescribedMotions in model m flagged with f:

```
var aprops = PrescribedMotion.AnimationBegin(m, f);
```

AnimationFinish() [static]

Description

Finishes a PrescribedMotion animation. This **MUST** be called to finish animating. This will restore nodal coordinates but will **not** perform a graphics update. Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating.

Arguments

No arguments

Return type

No return value

Example

To finish animating:

```
PrescribedMotion.AnimationFinish();
```

AnimationForward() [static]

Description

Moves forward one frame of a PrescribedMotion animation (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To move forward one frame of an animation:

```
PrescribedMotion.AnimationForward();
```

AnimationGetData() [static]

Description

Returns the animation data (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
end	real	End time
frame	integer	Current frame
frames	integer	Number of frames
rate	integer	Animation speed in fps
repeat	integer	Animation repeat (0=off, 1=on)
start	real	Start time
time	real	Current time
timestep	real	Timestep

Example

To get the current animation data:

```
PrescribedMotion.AnimationGetData();
```

AnimationPause() [static]

Description

Pauses playback of a PrescribedMotion animation. Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To pause playback of an animation:

```
PrescribedMotion.AnimationPause();
```

AnimationPlay() [static]

Description

Starts playback of a PrescribedMotion animation. Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.
This method should only be used from a script which implements a user interface so you can actually stop the animation! Don't forget to add a pause/stop button that calls PrescribedMotion.AnimationPause()!

Arguments

No arguments

Return type

No return value

Example

To start playback of an animation:

```
PrescribedMotion.AnimationPlay();
```

AnimationSetData(data*[object]*) [static]

Description

Sets the current animation data (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>data</td><td>object</td><td>data returned from PrescribedMotion.AnimationBegin() or PrescribedMotion.AnimationGetData()
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>end</td><td>real</td><td>End time</td></tr>
<tr><td>frame</td><td>integer</td><td>Current frame</td></tr>
<tr><td>frames</td><td>integer</td><td>Number of frames</td></tr>
<tr><td>rate</td><td>integer</td><td>Animation speed in fps</td></tr>
<tr><td>repeat</td><td>integer</td><td>Animation repeat (0=off, 1=on)</td></tr>
<tr><td>start</td><td>real</td><td>Start time</td></tr>
<tr><td>time</td><td>real</td><td>Current time</td></tr>
<tr><td>timestep</td><td>real</td><td>Timestep</td></tr>
</table>
Object has the following properties:</td></tr>
</table>

Return type

No return value

Example

To set the animation frame rate to 10 frames/sec:

```
data = PrescribedMotion.AnimationGetData();
data.rate = 10;
PrescribedMotion.AnimationSetData(data);
```

AnimationToEnd() [static]

Description

Moves to the end of a PrescribedMotion animation (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To move to the end of an animation:

```
PrescribedMotion.AnimationToEnd();
```

AnimationToStart() [static]

Description

Moves to the start of a PrescribedMotion animation (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To move to the start of an animation:

```
PrescribedMotion.AnimationToStart();
```

AnimationToTime() [static]

Description

Moves to a specific time in a PrescribedMotion animation (pausing animation first if required). Also see the PrescribedMotion.AnimationBegin() method which **MUST** be called before you start animating and the PrescribedMotion.AnimationFinish() method which **MUST** be called after you have finished animating.

Arguments

No arguments

Return type

No return value

Example

To move to time 28.0 in an animation:

```
PrescribedMotion.AnimationToTime(28.0);
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a boundary prescribed motion.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the boundary prescribed motion

Return type

No return value

Example

To associate comment c to the boundary prescribed motion b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the boundary prescribed motion

Arguments

No arguments

Return type

No return value

Example

To blank boundary prescribed motion b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary prescribed motions in model m:

```
PrescribedMotion.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed motions will be blanked in
flag	Flag	Flag set on the boundary prescribed motions that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary prescribed motions in model m flagged with f:

```
PrescribedMotion.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the boundary prescribed motion is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if boundary prescribed motion b is blanked:

```
if (b.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the boundary prescribed motion.

Arguments

Name	Type	Description

flag	Flag	Flag to clear on the boundary prescribed motion

Return type

No return value

Example

To clear flag f for boundary prescribed motion b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the boundary prescribed motion.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

PrescribedMotion object

Example

To copy boundary prescribed motion b into boundary prescribed motion z:

```
var z = b.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a boundary prescribed motion.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the boundary prescribed motion

Return type

No return value

Example

To detach comment c from the boundary prescribed motion b:

```
b.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for boundary prescribed motion. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for boundary prescribed motion b:

```
b.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first boundary prescribed motion in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first boundary prescribed motion in

Return type

PrescribedMotion object (or null if there are no boundary prescribed motions in the model).

Example

To get the first boundary prescribed motion in model m:

```
var b = PrescribedMotion.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free boundary prescribed motion label in the model. Also see PrescribedMotion.LastFreeLabel(), PrescribedMotion.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free boundary prescribed motion label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

PrescribedMotion label.

Example

To get the first free boundary prescribed motion label in model m:

```
var label = PrescribedMotion.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the boundary prescribed motions in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be flagged in
flag	Flag	Flag to set on the boundary prescribed motions

Return type

No return value

Example

To flag all of the boundary prescribed motions with flag f in model m:

```
PrescribedMotion.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the boundary prescribed motion is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the boundary prescribed motion

Return type

true if flagged, false if not.

Example

To check if boundary prescribed motion b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each boundary prescribed motion in the model.
Note that ForEach has been designed to make looping over boundary prescribed motions as fast as possible and so has some limitations.
Firstly, a single temporary PrescribedMotion object is created and on each function call it is updated with the current boundary prescribed motion data. This means that you should not try to store the PrescribedMotion object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new boundary prescribed motions inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions are in
func	function	Function to call for each boundary prescribed motion

extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the boundary prescribed motions in model m:

```
PrescribedMotion.ForEach(m, test);
function test(b)
{
// b is PrescribedMotion object
}
```

To call function test for all of the boundary prescribed motions in model m with optional object:

```
var data = { x:0, y:0 };
PrescribedMotion.ForEach(m, test, data);
function test(b, extra)
{
// b is PrescribedMotion object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of PrescribedMotion objects for all of the boundary prescribed motions in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary prescribed motions from

Return type

Array of PrescribedMotion objects

Example

To make an array of PrescribedMotion objects for all of the boundary prescribed motions in model m

```
var b = PrescribedMotion.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a boundary prescribed motion.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the boundary prescribed motion b:

```
var comm_array = b.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of PrescribedMotion objects for all of the flagged boundary prescribed motions in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary prescribed motions from
flag	Flag	Flag set on the boundary prescribed motions that you want to retrieve

Return type

Array of PrescribedMotion objects

Example

To make an array of PrescribedMotion objects for all of the boundary prescribed motions in model m flagged with f

```
var b = PrescribedMotion.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the PrescribedMotion object for a boundary prescribed motion ID.

Arguments

Name	Type	Description
Model	Model	Model to find the boundary prescribed motion in
number	integer	number of the boundary prescribed motion you want the PrescribedMotion object for

Return type

PrescribedMotion object (or null if boundary prescribed motion does not exist).

Example

To get the PrescribedMotion object for boundary prescribed motion 100 in model m

```
var b = PrescribedMotion.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a PrescribedMotion property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the PrescribedMotion.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	boundary prescribed motion property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if PrescribedMotion property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if PrescribedMotion property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this PrescribedMotion (*BOUNDARY_PRESCRIBED_MOTION_xxxx). **Note that a carriage return is not added**. See also PrescribedMotion.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for PrescribedMotion pm:

```
var key = pm.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the PrescribedMotion. **Note that a carriage return is not added**. See also PrescribedMotion.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for PrescribedMotion pm:

```
var cards = pm.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last boundary prescribed motion in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last boundary prescribed motion in

Return type

PrescribedMotion object (or null if there are no boundary prescribed motions in the model).

Example

To get the last boundary prescribed motion in model m:

```
var b = PrescribedMotion.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free boundary prescribed motion label in the model. Also see PrescribedMotion.FirstFreeLabel(), PrescribedMotion.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free boundary prescribed motion label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

PrescribedMotion label.

Example

To get the last free boundary prescribed motion label in model m:

```
var label = PrescribedMotion.LastFreeLabel(m);
```

Next()

Description

Returns the next boundary prescribed motion in the model.

Arguments

No arguments

Return type

PrescribedMotion object (or null if there are no more boundary prescribed motions in the model).

Example

To get the boundary prescribed motion in model m after boundary prescribed motion b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) boundary prescribed motion label in the model. Also see PrescribedMotion.FirstFreeLabel(), PrescribedMotion.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free boundary prescribed motion label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

PrescribedMotion label.

Example

To get the next free boundary prescribed motion label in model m:

```
var label = PrescribedMotion.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a boundary prescribed motion.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary prescribed motions from that model can be picked. If the argument is a Flag then only boundary prescribed motions that are flagged with *limit* can be selected. If omitted, or null, any boundary prescribed motions from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

PrescribedMotion object (or null if not picked)

Example

To pick a boundary prescribed motion from model m giving the prompt 'Pick boundary prescribed motion from screen':

```
var b = PrescribedMotion.Pick('Pick boundary prescribed motion from screen', m);
```

Previous()

Description

Returns the previous boundary prescribed motion in the model.

Arguments

No arguments

Return type

PrescribedMotion object (or null if there are no more boundary prescribed motions in the model).

Example

To get the boundary prescribed motion in model m before boundary prescribed motion b:

```
var b = b.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary prescribed motions in model m, from 1000000:

```
PrescribedMotion.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed motions will be renumbered in
flag	Flag	Flag set on the boundary prescribed motions that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary prescribed motions in model m flagged with f, from 1000000:

```
PrescribedMotion.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select boundary prescribed motions using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting boundary prescribed motions
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary prescribed motions from that model can be selected. If the argument is a Flag then only boundary prescribed motions that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any boundary prescribed motions can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of boundary prescribed motions selected or null if menu cancelled

Example

To select boundary prescribed motions from model m, flagging those selected with flag f, giving the prompt 'Select boundary prescribed motions':

```
PrescribedMotion.Select(f, 'Select boundary prescribed motions', m);
```

To select boundary prescribed motions, flagging those selected with flag f but limiting selection to boundary prescribed motions flagged with flag l, giving the prompt 'Select boundary prescribed motions':

```
PrescribedMotion.Select(f, 'Select boundary prescribed motions', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the boundary prescribed motion.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the boundary prescribed motion

Return type

No return value

Example

To set flag f for boundary prescribed motion b:

```
b.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the boundary prescribed motion. The boundary prescribed motion will be sketched until you either call PrescribedMotion.Unsketch(), PrescribedMotion.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed motion is sketched. If omitted redraw is true. If you want to sketch several boundary prescribed motions and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch boundary prescribed motion b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged boundary prescribed motions in the model. The boundary prescribed motions will be sketched until you either call PrescribedMotion.Unsketch(), PrescribedMotion.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary prescribed motions will be sketched in
flag	Flag	Flag set on the boundary prescribed motions that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed motions are sketched. If omitted redraw is true. If you want to sketch flagged boundary prescribed motions several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all boundary prescribed motions flagged with flag in model m:

```
PrescribedMotion.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing boundary prescribed motions should be counted. If false or omitted referenced but undefined boundary prescribed motions will also be included in the total.

Return type

number of boundary prescribed motions

Example

To get the total number of boundary prescribed motions in model m:

```
var total = PrescribedMotion.Total(m);
```

Unblank()

Description

Unblanks the boundary prescribed motion

Arguments

No arguments

Return type

No return value

Example

To unblank boundary prescribed motion b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary prescribed motions in model m:

```
PrescribedMotion.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged boundary prescribed motions will be unblanked in
flag	Flag	Flag set on the boundary prescribed motions that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary prescribed motions in model m flagged with f:

```
PrescribedMotion.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all boundary prescribed motions will be unset in
flag	Flag	Flag to unset on the boundary prescribed motions

Return type

No return value

Example

To unset the flag f on all the boundary prescribed motions in model m:

```
PrescribedMotion.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the boundary prescribed motion.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed motion is unsketched. If omitted redraw is true. If you want to unsketch several boundary prescribed motions and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch boundary prescribed motion b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all boundary prescribed motions.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed motions are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boundary prescribed motions in model m:

```
PrescribedMotion.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged boundary prescribed motions in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary prescribed motions will be unsketched in
flag	Flag	Flag set on the boundary prescribed motions that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the boundary prescribed motions are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boundary prescribed motions flagged with flag in model m:

```
PrescribedMotion.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

PrescribedMotion object.

Example

To check if PrescribedMotion property b.example is a parameter by using the PrescribedMotion.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for boundary prescribed motion. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for boundary prescribed motion b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this boundary prescribed motion.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for boundary prescribed motion b:

```
var xrefs = b.Xrefs();
```

---

toString()

Description

Creates a string containing the PrescribedMotion data in keyword format. Note that this contains the keyword header and the keyword cards. See also PrescribedMotion.Keyword() and PrescribedMotion.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for PrescribedMotion pm in keyword format

```
var s = pm.toString();
```

Spc class

The Spc class gives you access to define spc cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(type*[integer]*, Model*[Model]*, flag*[Flag]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(type*[constant]*, redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Spc constants

Constants for sketching mode

Name	Description
Spc.ROTATIONAL	Sketch rotational degrees of freedom.
Spc.TRANSLATIONAL	Sketch translational degrees of freedom.

Constants for suffix

Name	Description
Spc.NODE	SPC is *BOUNDARY_SPC_NODE.
Spc.SET	SPC is *BOUNDARY_SPC_SET.

Spc properties

Name	Type	Description
cid	integer	Coordinate system ID
dofrx	integer	Rotational constraint in local x direction
dofry	integer	Rotational constraint in local y direction
dofrz	integer	Rotational constraint in local z direction
dofx	integer	Translational constraint in local x direction
dofy	integer	Translational constraint in local y direction
dofz	integer	Translational constraint in local z direction
exists	logical	true if spc exists, false if referred to but not defined. (read only)
heading	string	Spc heading
id	logical	true if _ID option is set, false if not.
include	integer	The Include file number that the spc is in.
label	integer	Spc number.
model	integer	The Model number that the boundary SPC is in.
nid	integer	Node ID or node set ID
type	constant	The Spc type. Can be Spc.NODE or Spc.SET.

Properties for _BIRTH_DEATH option

Name	Type	Description
bd_flag	logical	true if _BIRTH_DEATH option is set, false if not
birth	real	Activation time for constraint
death	real	Deactivation time for constraint

Detailed Description

The Spc class allows you to create, modify, edit and manipulate spc cards. See the documentation below for more details.

Constructor

new Spc(Model*[Model]*, nid*[integer]*, cid*[integer]*, dofx*[integer]*, dofy*[integer]*, dofz*[integer]*, dofrx*[integer]*, dofry*[integer]*, dofrz*[integer]*, type*[constant]*, label (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Spc object.

Arguments

Name	Type	Description
Model	Model	Model that spc will be created in
nid	integer	Node ID or node set ID
cid	integer	Coordinate system ID
dofx	integer	Translational constraint in local x direction
dofy	integer	Translational constraint in local y direction
dofz	integer	Translational constraint in local z direction
dofrx	integer	Rotational constraint in local x direction
dofry	integer	Rotational constraint in local y direction
dofrz	integer	Rotational constraint in local z direction
type	constant	Specify the type of boundary spc (Can be Spc.NODE or Spc.SET)
label (optional)	integer	Spc number
heading (optional)	string	Title for the spc

Return type

Spc object

Example

To create a new boundary spc in model m with label 200, of type SET

```
var b = new Spc(m, 200, 0, 1, 0, 0, 1, 0, 0, Spc.SET, 200);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a boundary SPC.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the boundary SPC

Return type

No return value

Example

To associate comment c to the boundary SPC s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the boundary SPC

Arguments

No arguments

Return type

No return value

Example

To blank boundary SPC s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary SPCs will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary SPCs in model m:

```
Spc.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary SPCs will be blanked in

flag	Flag	Flag set on the boundary SPCs that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boundary SPCs in model m flagged with f:

```
Spc.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the boundary SPC is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if boundary SPC s is blanked:

```
if (s.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the boundary SPC.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the boundary SPC

Return type

No return value

Example

To clear flag f for boundary SPC s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the boundary SPC.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Spc object

Example

To copy boundary SPC s into boundary SPC z:

```
var z = s.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a boundary SPC.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the boundary SPC

Return type

No return value

Example

To detach comment c from the boundary SPC s:

```
s.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for boundary SPC. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for boundary SPC s:

```
s.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first boundary SPC in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first boundary SPC in

Return type

Spc object (or null if there are no boundary SPCs in the model).

Example

To get the first boundary SPC in model m:

```
var s = Spc.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free boundary SPC label in the model. Also see Spc.LastFreeLabel(), Spc.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free boundary SPC label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Spc label.

Example

To get the first free boundary SPC label in model m:

```
var label = Spc.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the boundary SPCs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all boundary SPCs will be flagged in
flag	Flag	Flag to set on the boundary SPCs

Return type

No return value

Example

To flag all of the boundary SPCs with flag f in model m:

```
Spc.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the boundary SPC is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the boundary SPC

Return type

true if flagged, false if not.

Example

To check if boundary SPC s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each boundary SPC in the model.
Note that ForEach has been designed to make looping over boundary SPCs as fast as possible and so has some limitations.
Firstly, a single temporary Spc object is created and on each function call it is updated with the current boundary SPC data. This means that you should not try to store the Spc object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new boundary SPCs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all boundary SPCs are in
func	function	Function to call for each boundary SPC
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the boundary SPCs in model m:

```
Spc.ForEach(m, test);
function test(s)
{
// s is Spc object
}
```

To call function test for all of the boundary SPCs in model m with optional object:

```
var data = { x:0, y:0 };
Spc.ForEach(m, test, data);
function test(s, extra)
{
// s is Spc object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Spc objects for all of the boundary SPCs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary SPCs from

Return type

Array of Spc objects

Example

To make an array of Spc objects for all of the boundary SPCs in model m

```
var s = Spc.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a boundary SPC.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the boundary SPC s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Spc objects for all of the flagged boundary SPCs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boundary SPCs from
flag	Flag	Flag set on the boundary SPCs that you want to retrieve

Return type

Array of Spc objects

Example

To make an array of Spc objects for all of the boundary SPCs in model m flagged with f

```
var s = Spc.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Spc object for a boundary SPC ID.

Arguments

Name	Type	Description
Model	Model	Model to find the boundary SPC in
number	integer	number of the boundary SPC you want the Spc object for

Return type

Spc object (or null if boundary SPC does not exist).

Example

To get the Spc object for boundary SPC 100 in model m

```
var s = Spc.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Spc property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Spc.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	boundary SPC property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Spc property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Spc property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this spc (*BOUNDARY_SPC_xxxx). **Note that a carriage return is not added**. See also Spc.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for spc s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the spc. **Note that a carriage return is not added**. See also Spc.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for spc s:

```
var cards = s.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last boundary SPC in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last boundary SPC in

Return type

Spc object (or null if there are no boundary SPCs in the model).

Example

To get the last boundary SPC in model m:

```
var s = Spc.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free boundary SPC label in the model. Also see Spc.FirstFreeLabel(), Spc.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free boundary SPC label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Spc label.

Example

To get the last free boundary SPC label in model m:

```
var label = Spc.LastFreeLabel(m);
```

Next()

Description

Returns the next boundary SPC in the model.

Arguments

No arguments

Return type

Spc object (or null if there are no more boundary SPCs in the model).

Example

To get the boundary SPC in model m after boundary SPC s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) boundary SPC label in the model. Also see Spc.FirstFreeLabel(), Spc.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free boundary SPC label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Spc label.

Example

To get the next free boundary SPC label in model m:

```
var label = Spc.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a boundary SPC.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary SPCs from that model can be picked. If the argument is a Flag then only boundary SPCs that are flagged with *limit* can be selected. If omitted, or null, any boundary SPCs from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Spc object (or null if not picked)

Example

To pick a boundary SPC from model m giving the prompt 'Pick boundary SPC from screen':

```
var s = Spc.Pick('Pick boundary SPC from screen', m);
```

Previous()

Description

Returns the previous boundary SPC in the model.

Arguments

No arguments

Return type

Spc object (or null if there are no more boundary SPCs in the model).

Example

To get the boundary SPC in model m before boundary SPC s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary SPCs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary SPCs in model m, from 1000000:

```
Spc.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boundary SPCs will be renumbered in
flag	Flag	Flag set on the boundary SPCs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boundary SPCs in model m flagged with f, from 1000000:

```
Spc.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select boundary SPCs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting boundary SPCs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boundary SPCs from that model can be selected. If the argument is a Flag then only boundary SPCs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any boundary SPCs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of boundary SPCs selected or null if menu cancelled

Example

To select boundary SPCs from model m, flagging those selected with flag f, giving the prompt 'Select boundary SPCs':

```
Spc.Select(f, 'Select boundary SPCs', m);
```

To select boundary SPCs, flagging those selected with flag f but limiting selection to boundary SPCs flagged with flag l, giving the prompt 'Select boundary SPCs':

```
Spc.Select(f, 'Select boundary SPCs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the boundary SPC.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the boundary SPC

Return type

No return value

Example

To set flag f for boundary SPC s:

```
s.SetFlag(f);
```

Sketch(type*[constant]*, redraw (optional)*[boolean]*)

Description

Sketches the Boundary SPC. The SPC will be sketched until you do a graphics update or delete the model

Arguments

Name	Type	Description
type	constant	Type of constraints to be drawn. Can be Spc.TRANSLATIONAL or Spc.ROTATIONAL.
redraw (optional)	boolean	If set to true (or omitted) the plot will be redrawn each time. If sketching a large number of items, efficiency will be gained by setting the argument to false for all but the last item sketched. The final call will redraw.

Return type

No return value

Example

To sketch SPC s - Translational constraint

```
s1.Sketch(Spc.TRANSLATIONAL, false);
s2.Sketch(Spc.TRANSLATIONAL, false);
s3.Sketch(Spc.TRANSLATIONAL, true);
```

SketchFlagged(type*[integer]*, Model*[Model]*, flag*[Flag]*) [static]

Description

Sketches all the flagged boundary SPCs in the model and update the plot. The SPCs will be sketched until you do a graphics update or delete the model.

Arguments

Name	Type	Description
type	integer	Type of constraints to be drawn. Can be Spc.TRANSLATIONAL or Spc.ROTATIONAL.
Model	Model	Model that all the flagged boundary SPCs will be sketched in
flag	Flag	Flag set on the boundary SPCs that you want to sketch

Return type

No return value

Example

To sketch translational SPCs flagged with f in model m and redraw

```
Spc.SketchFlagged(Spc.TRANSLATIONAL, m, f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing boundary SPCs should be counted. If false or omitted referenced but undefined boundary SPCs will also be included in the total.

Return type

number of boundary SPCs

Example

To get the total number of boundary SPCs in model m:

```
var total = Spc.Total(m);
```

Unblank()

Description

Unblanks the boundary SPC

Arguments

No arguments

Return type

No return value

Example

To unblank boundary SPC s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boundary SPCs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary SPCs in model m:

```
Spc.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged boundary SPCs will be unblanked in

flag	Flag	Flag set on the boundary SPCs that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boundary SPCs in model m flagged with f:

```
Spc.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the boundary SPCs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all boundary SPCs will be unset in
flag	Flag	Flag to unset on the boundary SPCs

Return type

No return value

Example

To unset the flag f on all the boundary SPCs in model m:

```
Spc.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the Spc.

Arguments

Name	Type	Description
redraw (optional)	boolean	If set to true (or omitted) the plot will be redrawn each time. If unsketching a large number of items, efficiency will be gained by setting the argument to false for all but the last item unsketched. The final call will redraw.

Return type

No return value

Example

To unsketch SPC s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*) [static]

Description

Unsketches all SPCs.

Arguments

Name	Type	Description
Model	Model	Model that all SPCs will be unblanked in

Return type

No return value

Example

To unsketch all SPCs in model m and redraw:

```
SPC.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsketches all flagged SPCs.

Arguments

Name	Type	Description
Model	Model	Model that all SPCs will be unsketched in
flag	Flag	Flag set on the SPCs that you want to unsketch

Return type

No return value

Example

To unsketch all SPCs in model m which are flagged with f and redraw:

```
SPC.UnsketchFlagged(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Spc object.

Example

To check if Spc property s.example is a parameter by using the Spc.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for boundary SPC. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for boundary SPC s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this boundary SPC.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for boundary SPC s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the spc data in keyword format. Note that this contains the keyword header and the keyword cards. See also Spc.Keyword() and Spc.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for spc s in keyword format

```
var str = s.toString();
```

Comment class

The Comment class gives you access to comment cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AddLine(Line content*[String or array of strings]*, Line number (optional)*[Integer]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DeleteLine(Line number*[Integer]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetLine(Line (optional)*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- ModifyLine(Line number*[Integer]*, New line content*[String]*)
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Comment constants

Constants for Comment anchor_mode types

Name	Description
Comment.MULTIPLE	The *COMMENT is associated with all cards in the next block of keywords.
Comment.SINGLE	The *COMMENT is associated with just the one immediately following keyword.

Comment properties

Name	Type	Description

anchor_mode	integer	Anchor mode. Can be Comment.SINGLE, Comment.MULTIPLE.
exists	logical	True if comment exists, false if referred to but not defined (read only).
header	string	The header of the comment, or empty if the comment has no header.
include	integer	The Include file number that the comment is in.
model	integer	The Model number that the comment is in.
nlines	integer	Number of lines in the comment.

Detailed Description

The Comment class allows you to create, modify, edit and manipulate comment cards. See the documentation below for more details.

Constructor

new Comment(Model*[Model]*, Header (optional)*[string]*, Mode (optional)*[constant]*)

Description

Create a new Comment object.

Arguments

Name	Type	Description
Model	Model	Model that comment will be created in
Header (optional)	string	Comment number
Mode (optional)	constant	Anchor: single or multiple

Return type

Comment object

Example

To create a new comment in model m with header "My header", and multiple anchor:

```
var c = new Comment(m, "My header", Comment.MULTIPLE);
```

To create a new comment in model m without header, and single anchor:

```
var c = new Comment(m);
```

Details of functions

AddLine(Line content*[String or array of strings]*, Line number (optional)*[Integer]*)

Description

Adds a line, or an array of lines, to a comment object.

Arguments

Name	Type	Description

Line content	String or array of strings	String that will be added to a line
Line number (optional)	Integer	0: First line, 1: Second line, etc. If array of lines has been passed in the first argument, the first line of the array will be inserted in the line number specified in second argument, the second line of the array will be inserted in the following line number, etc. If that line already exists, that line and rest of them below will be shifted down. If greater than number of existing lines, blank lines will be added. If lower than 0, not valid argument. If no argument, the line(s) will be appended at the end.

Return type

no return value

Example

To add a new line in the second row of comment c:

```
var str = c.AddLine("New line", 1);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse comment c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the comment.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the comment

Return type

No return value

Example

To clear flag f for comment c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the comment.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Comment object

Example

To copy comment c into comment z:

```
var z = c.Copy();
```

DeleteLine(Line number*[Integer]*)

Description

Deletes a line of a comment.

Arguments

Name	Type	Description
Line number	Integer	Line number to delete (starting at 0). The following lines will be shifted up.

Return type

no return value

Example

To delete the line in the second row of comment c:

```
var str = c.DeleteLine(1);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit comment c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for comment. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for comment c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first comment in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first comment in

Return type

Comment object (or null if there are no comments in the model).

Example

To get the first comment in model m:

```
var c = Comment.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the comments in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all comments will be flagged in
flag	Flag	Flag to set on the comments

Return type

No return value

Example

To flag all of the comments with flag f in model m:

```
Comment.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the comment is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the comment

Return type

true if flagged, false if not.

Example

To check if comment c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each comment in the model.
Note that ForEach has been designed to make looping over comments as fast as possible and so has some limitations.
Firstly, a single temporary Comment object is created and on each function call it is updated with the current comment data. This means that you should not try to store the Comment object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new comments inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all comments are in
func	function	Function to call for each comment
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the comments in model m:

```
Comment.ForEach(m, test);
function test(c)
{
// c is Comment object
}
```

To call function test for all of the comments in model m with optional object:

```
var data = { x:0, y:0 };
Comment.ForEach(m, test, data);
function test(c, extra)
{
// c is Comment object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Comment objects for all of the comments in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get comments from

Return type

Array of Comment objects

Example

To make an array of Comment objects for all of the comments in model m

```
var c = Comment.GetAll(m);
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of Comment objects for all of the flagged comments in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get comments from
flag	Flag	Flag set on the comments that you want to retrieve

Return type

Array of Comment objects

Example

To make an array of Comment objects for all of the comments in model m flagged with f

```
var c = Comment.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Comment object for a comment ID.

Arguments

Name	Type	Description
Model	Model	Model to find the comment in
number	integer	number of the comment you want the Comment object for

Return type

Comment object (or null if comment does not exist).

Example

To get the Comment object for comment 100 in model m

```
var c = Comment.GetFromID(m, 100);
```

GetLine(Line (optional)*[integer]*)

Description

Extracts the lines (the strings) from a comment object.

Arguments

Name	Type	Description
Line (optional)	integer	Line number to be extracted. Default value: 0 (first line)

Return type

String (or null if no lines in the comment and not argument passed)

Example

To extract the first line of comment c:

```
var str = c.GetLine();
```

GetParameter(prop*[string]*)

Description

Checks if a Comment property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Comment.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	comment property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Comment property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Comment property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this comment (*COMMENT) and the header of the comment if there is one. **Note that a carriage return is not added**. See also Comment.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for comment c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the comment. **Note that a carriage return is not added**. See also Comment.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for comment c:

```
var cards = c.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last comment in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last comment in

Return type

Comment object (or null if there are no comments in the model).

Example

To get the last comment in model m:

```
var c = Comment.Last(m);
```

ModifyLine(Line number*[Integer]*, New line content*[String]*)

Description

Modifies the content of a line in a comment.

Arguments

Name	Type	Description
Line number	Integer	Line number to modify (starting at 0)
New line content	String	String that replaces the existing one in a line

Return type

no return value

Example

To modify the line in the second row of comment c:

```
var str = c.ModifyLine(1, "Modified line");
```

Next()

Description

Returns the next comment in the model.

Arguments

No arguments

Return type

Comment object (or null if there are no more comments in the model).

Example

To get the comment in model m after comment c:

```
var c = c.Next();
```

Previous()

Description

Returns the previous comment in the model.

Arguments

No arguments

Return type

Comment object (or null if there are no more comments in the model).

Example

To get the comment in model m before comment c:

```
var c = c.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select comments using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting comments
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only comments from that model can be selected. If the argument is a Flag then only comments that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any comments can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of comments selected or null if menu cancelled

Example

To select comments from model m, flagging those selected with flag f, giving the prompt 'Select comments':

```
Comment.Select(f, 'Select comments', m);
```

To select comments, flagging those selected with flag f but limiting selection to comments flagged with flag l, giving the prompt 'Select comments':

```
Comment.Select(f, 'Select comments', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the comment.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the comment

Return type

No return value

Example

To set flag f for comment c:

```
c.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of comments in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing comments should be counted. If false or omitted referenced but undefined comments will also be included in the total.

Return type

number of comments

Example

To get the total number of comments in model m:

```
var total = Comment.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the comments in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all comments will be unset in
flag	Flag	Flag to unset on the comments

Return type

No return value

Example

To unset the flag f on all the comments in model m:

```
Comment.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Comment object.

Example

To check if Comment property c.example is a parameter by using the Comment.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for comment. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for comment c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this comment.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for comment c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the comment data in keyword format. Note that this contains the keyword header and the keyword cards. See also Comment.Keyword() and Comment.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for comment c in keyword format

```
var s = c.toString();
```

ExtraNodes class

The ExtraNodes class gives you access to constrained extra nodes cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ExtraNodes constants

Name	Description
ExtraNodes.NODE	CNST is *CONSTRAINED_EXTRA_NODES_NODE.
ExtraNodes.SET	CNST is *CONSTRAINED_EXTRA_NODES_SET.

ExtraNodes properties

Name	Type	Description
exists	logical	true if constrained extra nodes exists, false if referred to but not defined (read only)
id	integer	Node ID or node set ID (not internal label)
iflag	logical	Flag for adding node mass inertia to PART_INERTIA
include	integer	The Include file number that the constrained extra nodes is in.
label	integer	The label the constrained extra nodes has in PRIMER (read only)
model	integer	The Model number that the constrained extra node is in.
option	constant	The Constrained Extra Nodes option. Can be ExtraNodes.NODE or ExtraNodes.SET.
pid	integer	Part ID of rigid body.

Detailed Description

The ExtraNodes class allows you to create, modify, edit and manipulate constrained extra nodes cards. See the documentation below for more details.

Constructor

new ExtraNodes(Model*[Model]*, option*[constant]*, pid*[integer]*, id*[integer]*, iflag*[boolean]*)

Description

Create a new ExtraNodes object.

Arguments

Name	Type	Description
Model	Model	Model that constrained extra nodes will be created in
option	constant	Specify the type of constrained extra nodes. Can be ExtraNodes.NODE or ExtraNodes.SET)
pid	integer	Part ID of rigid body
id	integer	Node node ID or node set ID
iflag	boolean	Flag for adding node mass inertia to PART_INERTIA

Return type

ExtraNodes object

Example

To create a new constrained extra nodes in model m, of type SET, with part 9, node set 18 and iflag 0

```
var e = new ExtraNodes(m, ExtraNodes.SET, 9, 18, 0);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a constrained extra node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the constrained extra node

Return type

No return value

Example

To associate comment c to the constrained extra node en:

```
en.AssociateComment(c);
```

Blank()

Description

Blanks the constrained extra node

Arguments

No arguments

Return type

No return value

Example

To blank constrained extra node en:

```
en.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained extra nodes will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained extra nodes in model m:

```
ExtraNodes.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained extra nodes will be blanked in
flag	Flag	Flag set on the constrained extra nodes that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained extra nodes in model m flagged with f:

```
ExtraNodes.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the constrained extra node is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if constrained extra node en is blanked:

```
if (en.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse constrained extra node en:

```
en.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the constrained extra node.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the constrained extra node

Return type

No return value

Example

To clear flag f for constrained extra node en:

```
en.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the constrained extra node.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ExtraNodes object

Example

To copy constrained extra node en into constrained extra node z:

```
var z = en.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a constrained extra nodes card.

Arguments

Name	Type	Description
Model	Model	Model that the constrained extra nodes card will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ExtraNodes object (or null if not made)

Example

To start creating a constrained extra nodes card in model m:

```
var e = ExtraNodes.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a constrained extra node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the constrained extra node

Return type

No return value

Example

To detach comment c from the constrained extra node en:

```
en.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit constrained extra node en:

```
en.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for constrained extra node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for constrained extra node en:

```
en.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first constrained extra node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first constrained extra node in

Return type

ExtraNodes object (or null if there are no constrained extra nodes in the model).

Example

To get the first constrained extra node in model m:

```
var en = ExtraNodes.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the constrained extra nodes in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all constrained extra nodes will be flagged in
flag	Flag	Flag to set on the constrained extra nodes

Return type

No return value

Example

To flag all of the constrained extra nodes with flag f in model m:

```
ExtraNodes.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the constrained extra node is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the constrained extra node

Return type

true if flagged, false if not.

Example

To check if constrained extra node en has flag f set on it:

```
if (en.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each constrained extra node in the model.
Note that ForEach has been designed to make looping over constrained extra nodes as fast as possible and so has some limitations.
Firstly, a single temporary ExtraNodes object is created and on each function call it is updated with the current constrained extra node data. This means that you should not try to store the ExtraNodes object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new constrained extra nodes inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all constrained extra nodes are in
func	function	Function to call for each constrained extra node
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the constrained extra nodes in model m:

```
ExtraNodes.ForEach(m, test);
function test(en)
{
// en is ExtraNodes object
}
```

To call function test for all of the constrained extra nodes in model m with optional object:

```
var data = { x:0, y:0 };
ExtraNodes.ForEach(m, test, data);
function test(en, extra)
{
// en is ExtraNodes object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of ExtraNodes objects for all of the constrained extra nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained extra nodes from

Return type

Array of ExtraNodes objects

Example

To make an array of ExtraNodes objects for all of the constrained extra nodes in model m

```
var en = ExtraNodes.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a constrained extra node.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the constrained extra node en:

```
var comm_array = en.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of ExtraNodes objects for all of the flagged constrained extra nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained extra nodes from
flag	Flag	Flag set on the constrained extra nodes that you want to retrieve

Return type

Array of ExtraNodes objects

Example

To make an array of ExtraNodes objects for all of the constrained extra nodes in model m flagged with f

```
var en = ExtraNodes.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ExtraNodes object for a constrained extra node ID.

Arguments

Name	Type	Description
Model	Model	Model to find the constrained extra node in
number	integer	number of the constrained extra node you want the ExtraNodes object for

Return type

ExtraNodes object (or null if constrained extra node does not exist).

Example

To get the ExtraNodes object for constrained extra node 100 in model m

```
var en = ExtraNodes.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a ExtraNodes property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ExtraNodes.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	constrained extra node property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ExtraNodes property en.example is a parameter:

```
Options.property_parameter_names = true;
if (en.GetParameter(en.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ExtraNodes property en.example is a parameter by using the GetParameter method:

```
if (en.ViewParameters().GetParameter(en.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this constrained extra nodes (*CONSTRAINED_EXTRA_NODES). **Note that a carriage return is not added**. See also ExtraNodes.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for constrained extra nodes e:

```
var key = e.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the constrained extra nodes. **Note that a carriage return is not added**. See also ExtraNodes.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for constrained extra nodes e:

```
var cards = e.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last constrained extra node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last constrained extra node in

Return type

ExtraNodes object (or null if there are no constrained extra nodes in the model).

Example

To get the last constrained extra node in model m:

```
var en = ExtraNodes.Last(m);
```

Next()

Description

Returns the next constrained extra node in the model.

Arguments

No arguments

Return type

ExtraNodes object (or null if there are no more constrained extra nodes in the model).

Example

To get the constrained extra node in model m after constrained extra node en:

```
var en = en.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a constrained extra node.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained extra nodes from that model can be picked. If the argument is a Flag then only constrained extra nodes that are flagged with *limit* can be selected. If omitted, or null, any constrained extra nodes from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

ExtraNodes object (or null if not picked)

Example

To pick a constrained extra node from model m giving the prompt 'Pick constrained extra node from screen':

```
var en = ExtraNodes.Pick('Pick constrained extra node from screen', m);
```

Previous()

Description

Returns the previous constrained extra node in the model.

Arguments

No arguments

Return type

ExtraNodes object (or null if there are no more constrained extra nodes in the model).

Example

To get the constrained extra node in model m before constrained extra node en:

```
var en = en.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select constrained extra nodes using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting constrained extra nodes
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained extra nodes from that model can be selected. If the argument is a Flag then only constrained extra nodes that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any constrained extra nodes can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of constrained extra nodes selected or null if menu cancelled

Example

To select constrained extra nodes from model m, flagging those selected with flag f, giving the prompt 'Select constrained extra nodes':

```
ExtraNodes.Select(f, 'Select constrained extra nodes', m);
```

To select constrained extra nodes, flagging those selected with flag f but limiting selection to constrained extra nodes flagged with flag l, giving the prompt 'Select constrained extra nodes':

```
ExtraNodes.Select(f, 'Select constrained extra nodes', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the constrained extra node.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the constrained extra node

Return type

No return value

Example

To set flag f for constrained extra node en:

```
en.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the constrained extra node. The constrained extra node will be sketched until you either call ExtraNodes.Unsketch(), ExtraNodes.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained extra node is sketched. If omitted redraw is true. If you want to sketch several constrained extra nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch constrained extra node en:

```
en.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged constrained extra nodes in the model. The constrained extra nodes will be sketched until you either call ExtraNodes.Unsketch(), ExtraNodes.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained extra nodes will be sketched in
flag	Flag	Flag set on the constrained extra nodes that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the constrained extra nodes are sketched. If omitted redraw is true. If you want to sketch flagged constrained extra nodes several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all constrained extra nodes flagged with flag in model m:

```
ExtraNodes.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing constrained extra nodes should be counted. If false or omitted referenced but undefined constrained extra nodes will also be included in the total.

Return type

number of constrained extra nodes

Example

To get the total number of constrained extra nodes in model m:

```
var total = ExtraNodes.Total(m);
```

Unblank()

Description

Unblanks the constrained extra node

Arguments

No arguments

Return type

No return value

Example

To unblank constrained extra node en:

```
en.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the constrained extra nodes in the model.

Arguments

Name	Type	Description

Model	Model	Model that all constrained extra nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained extra nodes in model m:

```
ExtraNodes.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged constrained extra nodes will be unblanked in
flag	Flag	Flag set on the constrained extra nodes that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained extra nodes in model m flagged with f:

```
ExtraNodes.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all constrained extra nodes will be unset in
flag	Flag	Flag to unset on the constrained extra nodes

Return type

No return value

Example

To unset the flag f on all the constrained extra nodes in model m:

```
ExtraNodes.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the constrained extra node.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained extra node is unsketched. If omitted redraw is true. If you want to unsketch several constrained extra nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch constrained extra node en:

```
en.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all constrained extra nodes.

Arguments

Name	Type	Description
Model	Model	Model that all constrained extra nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the constrained extra nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained extra nodes in model m:

```
ExtraNodes.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged constrained extra nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained extra nodes will be unsketched in
flag	Flag	Flag set on the constrained extra nodes that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the constrained extra nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained extra nodes flagged with flag in model m:

```
ExtraNodes.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ExtraNodes object.

Example

To check if ExtraNodes property en.example is a parameter by using the ExtraNodes.GetParameter() method:

```
if (en.ViewParameters().GetParameter(en.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for constrained extra node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for constrained extra node en:

```
en.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this constrained extra node.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for constrained extra node en:

```
var xrefs = en.Xrefs();
```

---

toString()

Description

Creates a string containing the constrained extra nodes data in keyword format. Note that this contains the keyword header and the keyword cards. See also ExtraNodes.Keyword() and ExtraNodes.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for constrained extra nodes e in keyword format

```
var s = e.toString();
```

GeneralizedWeld (Gwld) class

The GeneralizedWeld class gives you access to constrained generalized weld cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetCombinedData(index*[integer]*)
- GetComments()
- GetCrossFilletData(index*[integer]*)
- GetFailureData() [deprecated]
- GetNodalPair() [deprecated]
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetCombinedData(index*[integer]*, data*[Array of numbers]*)
- SetCrossFilletData(index*[integer]*, data*[Array of numbers]*)
- SetFailureData() [deprecated]
- SetFlag(flag*[Flag]*)

- SetNodalPair() [deprecated]
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

GeneralizedWeld constants

Name	Description
GeneralizedWeld.BUTT	GeneralizedWeld is *CONSTRAINED_GENERALIZED_WELD_BUTT.
GeneralizedWeld.COMBINED	GeneralizedWeld is *CONSTRAINED_GENERALIZED_WELD_COMBINED.
GeneralizedWeld.CROSS_ FILLET	GeneralizedWeld is *CONSTRAINED_GENERALIZED_WELD_CROSS_ FILLET.
GeneralizedWeld.FILLET	GeneralizedWeld is *CONSTRAINED_GENERALIZED_WELD_FILLET.
GeneralizedWeld.SPOT	GeneralizedWeld is *CONSTRAINED_GENERALIZED_WELD_SPOT.

GeneralizedWeld properties

Name	Type	Description
a	real	Width of fillet (GeneralizedWeld.FILLET, GeneralizedWeld.CROSS_FILLET)
alpha	real	Weld angle (GeneralizedWeld.FILLET, GeneralizedWeld.CROSS_FILLET)
beta	real	Failure parameter (GeneralizedWeld.FILLET, GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_FILLET)
cid	integer	Coordinate System ID.
d	real	Thickness of weld (GeneralizedWeld.BUTT)
epsf	real	Effective plastic strain at failure (GeneralizedWeld.SPOT, GeneralizedWeld.FILLET, GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_FILLET)
exists	logical	true if gwld exists, false if referred to but not defined. (read only)
filter	integer	Number of force vectors saved for filtering.
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the gwld is in.
l	real	Length of weld (GeneralizedWeld.FILLET, GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_ FILLET)
label	integer	Constrained Generalized weld number.
lt	real	Transverse length (GeneralizedWeld.BUTT)
m	real	Exponent for shear force (GeneralizedWeld.SPOT)
model	integer	The Model number that the generalized weld is in.
n	real	Exponent for normal force (GeneralizedWeld.SPOT)
npr	integer	Number of individual nodal pairs in cross fillet and combined weld.
nprt	integer	Printout option.
nsid	integer	Set Node Set ID.
option	constant	GeneralizedWeld type. Can be GeneralizedWeld.SPOT, GeneralizedWeld.FILLET, GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_FILLET, GeneralizedWeld.COMBINED

sigf	real	Stress at failure (GeneralizedWeld.FILLET)
sigy	real	Stress at failure (GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_FILLET)
sn	real	Normal force at failure (GeneralizedWeld.SPOT)
ss	real	Shear force at faliure (GeneralizedWeld.SPOT)
tfail	real	Failure time for constraint set (GeneralizedWeld.SPOT, GeneralizedWeld.FILLET, GeneralizedWeld.BUTT, GeneralizedWeld.CROSS_FILLET)
w	real	Width of flange (GeneralizedWeld.FILLET, GeneralizedWeld.CROSS_FILLET)
wid	integer	Constrained Generalized weld number (identical to label).
window	real	Filter time window.

Detailed Description

The GeneralizedWeld class allows you to create, modify, edit and manipulate generalized weld cards. See the documentation below for more details.

For convenience "Gwld" can also be used as the class name instead of "GeneralizedWeld".

Constructor

new GeneralizedWeld(Model*[Model]*, option*[constant]*, nsid*[integer]*, cid (optional)*[integer]*, filter (optional)*[integer]*, window (optional)*[real]*, npr (optional)*[integer]*, nprt (optional)*[integer]*, wid (optional)*[integer]*)

Description

Create a new GeneralizedWeld object.

Arguments

Name	Type	Description
Model	Model	Model that gwld will be created in
option	constant	Constrained generalized weld type (any).
nsid	integer	Set Node Set ID.
cid (optional)	integer	Coordinate System ID.
filter (optional)	integer	Number of force vectors saved for filtering.
window (optional)	real	Filter time window.
npr (optional)	integer	Number of individual nodal pairs in cross fillet and combined weld.
nprt (optional)	integer	Printout option.
wid (optional)	integer	Constrained Generalized weld number.

Return type

GeneralizedWeld object

Example

To create a new gwld 1000 of type SPOT in model m with specifiction: nsid, cid, filter, window, nprt are 91, 92, 81, 0.5, 82 respectively

```
var w = new GeneralizedWeld(m, GeneralizedWeld.SPOT, 91, 92, 81, 0.5, 82, 1000);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a generalized weld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the generalized weld

Return type

No return value

Example

To associate comment c to the generalized weld gw:

```
gw.AssociateComment(c);
```

Blank()

Description

Blanks the generalized weld

Arguments

No arguments

Return type

No return value

Example

To blank generalized weld gw:

```
gw.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the generalized welds in model m:

```
GeneralizedWeld.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged generalized welds will be blanked in
flag	Flag	Flag set on the generalized welds that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the generalized welds in model m flagged with f:

```
GeneralizedWeld.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the generalized weld is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if generalized weld gw is blanked:

```
if (gw.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse generalized weld gw:

```
gw.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the generalized weld.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the generalized weld

Return type

No return value

Example

To clear flag f for generalized weld gw:

```
gw.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the generalized weld.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

GeneralizedWeld object

Example

To copy generalized weld gw into generalized weld z:

```
var z = gw.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a gwld.

Arguments

Name	Type	Description
Model	Model	Model that the gwld will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

GeneralizedWeld object (or null if not made)

Example

To start creating a generalized weld in model m:

```
var gw = GeneralizedWeld.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a generalized weld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the generalized weld

Return type

No return value

Example

To detach comment c from the generalized weld gw:

```
gw.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit generalized weld gw:

```
gw.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for generalized weld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for generalized weld gw:

```
gw.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first generalized weld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first generalized weld in

Return type

GeneralizedWeld object (or null if there are no generalized welds in the model).

Example

To get the first generalized weld in model m:

```
var gw = GeneralizedWeld.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free generalized weld label in the model. Also see GeneralizedWeld.LastFreeLabel(), GeneralizedWeld.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free generalized weld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

GeneralizedWeld label.

Example

To get the first free generalized weld label in model m:

```
var label = GeneralizedWeld.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the generalized welds in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be flagged in
flag	Flag	Flag to set on the generalized welds

Return type

No return value

Example

To flag all of the generalized welds with flag f in model m:

```
GeneralizedWeld.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the generalized weld is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the generalized weld

Return type

true if flagged, false if not.

Example

To check if generalized weld gw has flag f set on it:

```
if (gw.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each generalized weld in the model.
Note that ForEach has been designed to make looping over generalized welds as fast as possible and so has some limitations.
Firstly, a single temporary GeneralizedWeld object is created and on each function call it is updated with the current generalized weld data. This means that you should not try to store the GeneralizedWeld object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new generalized welds inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds are in
func	function	Function to call for each generalized weld
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the generalized welds in model m:

```
GeneralizedWeld.ForEach(m, test);
function test(gw)
{
// gw is GeneralizedWeld object
}
```

To call function test for all of the generalized welds in model m with optional object:

```
var data = { x:0, y:0 };
GeneralizedWeld.ForEach(m, test, data);
function test(gw, extra)
{
// gw is GeneralizedWeld object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of GeneralizedWeld objects for all of the generalized welds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get generalized welds from

Return type

Array of GeneralizedWeld objects

Example

To make an array of GeneralizedWeld objects for all of the generalized welds in model m

```
var gw = GeneralizedWeld.GetAll(m);
```

GetCombinedData(index*[integer]*)

Description

Returns the combined data for a specific nodal pair as an array.

Arguments

Name	Type	Description
index	integer	Index you want the data for. **Note that indices start at 0**.

Return type

An array containing the data (tfail, epsf, sigy, beta, l, w, a, alpha, nodea, nodeb, ncid, wtyp).

Example

To get the data for the 3rd node pair for generalized weld gw:

```
var data = gw.GetCombinedData(2);
```

GetComments()

Description

Extracts the comments associated to a generalized weld.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the generalized weld gw:

```
var comm_array = gw.GetComments();
```

GetCrossFilletData(index*[integer]*)

Description

Returns the cross fillet data for a specific nodal pair as an array.

Arguments

Name	Type	Description
index	integer	Index you want the data for. **Note that indices start at 0**.

Return type

An array containing the data (nodea, nodeb, ncid).

Example

To get the data for the 3rd node pair for generalized weld gw:

```
var data = gw.GetCrossFilletData(2);
```

GetFailureData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Access the properties directly or use GeneralizedWeld.GetCombinedData() for GeneralizedWeld.COMBINED instead.

Arguments

No arguments

Return type

No return value

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of GeneralizedWeld objects for all of the flagged generalized welds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get generalized welds from
flag	Flag	Flag set on the generalized welds that you want to retrieve

Return type

Array of GeneralizedWeld objects

Example

To make an array of GeneralizedWeld objects for all of the generalized welds in model m flagged with f

```
var gw = GeneralizedWeld.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the GeneralizedWeld object for a generalized weld ID.

Arguments

Name	Type	Description
Model	Model	Model to find the generalized weld in
number	integer	number of the generalized weld you want the GeneralizedWeld object for

Return type

GeneralizedWeld object (or null if generalized weld does not exist).

Example

To get the GeneralizedWeld object for generalized weld 100 in model m

```
var gw = GeneralizedWeld.GetFromID(m, 100);
```

GetNodalPair() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use GeneralizedWeld.GetCombinedData() for GeneralizedWeld.COMBINED or GeneralizedWeld.GetCrossFilletData() for GeneralizedWeld.CROSS_FILLET instead.

Arguments

No arguments

Return type

No return value

GetParameter(prop*[string]*)

Description

Checks if a GeneralizedWeld property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the GeneralizedWeld.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	generalized weld property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if GeneralizedWeld property gw.example is a parameter:

```
Options.property_parameter_names = true;
if (gw.GetParameter(gw.example) ) do_something...
Options.property_parameter_names = false;
```

To check if GeneralizedWeld property gw.example is a parameter by using the GetParameter method:

```
if (gw.ViewParameters().GetParameter(gw.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this gwld (*CONSTRAINED_GENERALIZED_WELD_xxxx). **Note that a carriage return is not added**. See also GeneralizedWeld.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for generalized weld gw:

```
var key = gw.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the gwld. **Note that a carriage return is not added**. See also GeneralizedWeld.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for generalized weld gw:

```
var cards = gw.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last generalized weld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last generalized weld in

Return type

GeneralizedWeld object (or null if there are no generalized welds in the model).

Example

To get the last generalized weld in model m:

```
var gw = GeneralizedWeld.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free generalized weld label in the model. Also see GeneralizedWeld.FirstFreeLabel(), GeneralizedWeld.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free generalized weld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

GeneralizedWeld label.

Example

To get the last free generalized weld label in model m:

```
var label = GeneralizedWeld.LastFreeLabel(m);
```

Next()

Description

Returns the next generalized weld in the model.

Arguments

No arguments

Return type

GeneralizedWeld object (or null if there are no more generalized welds in the model).

Example

To get the generalized weld in model m after generalized weld gw:

```
var gw = gw.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) generalized weld label in the model. Also see GeneralizedWeld.FirstFreeLabel(), GeneralizedWeld.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free generalized weld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

GeneralizedWeld label.

Example

To get the next free generalized weld label in model m:

```
var label = GeneralizedWeld.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a generalized weld.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only generalized welds from that model can be picked. If the argument is a Flag then only generalized welds that are flagged with *limit* can be selected. If omitted, or null, any generalized welds from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

GeneralizedWeld object (or null if not picked)

Example

To pick a generalized weld from model m giving the prompt 'Pick generalized weld from screen':

```
var gw = GeneralizedWeld.Pick('Pick generalized weld from screen', m);
```

Previous()

Description

Returns the previous generalized weld in the model.

Arguments

No arguments

Return type

GeneralizedWeld object (or null if there are no more generalized welds in the model).

Example

To get the generalized weld in model m before generalized weld gw:

```
var gw = gw.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the generalized welds in model m, from 1000000:

```
GeneralizedWeld.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged generalized welds will be renumbered in
flag	Flag	Flag set on the generalized welds that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the generalized welds in model m flagged with f, from 1000000:

```
GeneralizedWeld.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select generalized welds using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting generalized welds
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only generalized welds from that model can be selected. If the argument is a Flag then only generalized welds that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any generalized welds can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of generalized welds selected or null if menu cancelled

Example

To select generalized welds from model m, flagging those selected with flag f, giving the prompt 'Select generalized welds':

```
GeneralizedWeld.Select(f, 'Select generalized welds', m);
```

To select generalized welds, flagging those selected with flag f but limiting selection to generalized welds flagged with flag l, giving the prompt 'Select generalized welds':

```
GeneralizedWeld.Select(f, 'Select generalized welds', l);
```

SetCombinedData(index*[integer]*, data*[Array of numbers]*)

Description

Sets the combined data for a specific nodal pair.

Arguments

Name	Type	Description
index	integer	Index you want to set the data for. **Note that indices start at 0**.
data	Array of numbers	Array containing the data. The array length should be 12 (tfail, epsf, sigy, beta, l, w, a, alpha, nodea, nodeb, ncid, wtyp)

Return type

No return value.

Example

To set the data for the 3rd nodal pair for generalized weld gw to the values in array adata:

```
gw.SetCombinedData(2, adata);
```

SetCrossFilletData(index*[integer]*, data*[Array of numbers]*)

Description

Sets the cross fillet data for a specific nodal pair.

Arguments

Name	Type	Description
index	integer	Index you want to set the data for. **Note that indices start at 0**.
data	Array of numbers	Array containing the data. The array length should be 3 (nodea, nodeb, ncid)

Return type

No return value.

Example

To set the data for the 3rd nodal pair for generalized weld gw to the values in array adata:

```
gw.SetCrossFilletData(2, adata);
```

SetFailureData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Access the properties directly or use GeneralizedWeld.SetCombinedData() for GeneralizedWeld.COMBINED instead.

Arguments

No arguments

Return type

No return value

SetFlag(flag*[Flag]*)

Description

Sets a flag on the generalized weld.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the generalized weld

Return type

No return value

Example

To set flag f for generalized weld gw:

```
gw.SetFlag(f);
```

SetNodalPair() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use GeneralizedWeld.SetCombinedData() for GeneralizedWeld.COMBINED or GeneralizedWeld.SetCrossFilletData() for GeneralizedWeld.CROSS_FILLET instead.

Arguments

No arguments

Return type

No return value

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the generalized weld. The generalized weld will be sketched until you either call GeneralizedWeld.Unsketch(), GeneralizedWeld.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the generalized weld is sketched. If omitted redraw is true. If you want to sketch several generalized welds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch generalized weld gw:

```
gw.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged generalized welds in the model. The generalized welds will be sketched until you either call GeneralizedWeld.Unsketch(), GeneralizedWeld.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged generalized welds will be sketched in
flag	Flag	Flag set on the generalized welds that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the generalized welds are sketched. If omitted redraw is true. If you want to sketch flagged generalized welds several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all generalized welds flagged with flag in model m:

```
GeneralizedWeld.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing generalized welds should be counted. If false or omitted referenced but undefined generalized welds will also be included in the total.

Return type

number of generalized welds

Example

To get the total number of generalized welds in model m:

```
var total = GeneralizedWeld.Total(m);
```

Unblank()

Description

Unblanks the generalized weld

Arguments

No arguments

Return type

No return value

Example

To unblank generalized weld gw:

```
gw.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the generalized welds in model m:

```
GeneralizedWeld.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged generalized welds will be unblanked in

flag	Flag	Flag set on the generalized welds that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the generalized welds in model m flagged with f:

```
GeneralizedWeld.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all generalized welds will be unset in
flag	Flag	Flag to unset on the generalized welds

Return type

No return value

Example

To unset the flag f on all the generalized welds in model m:

```
GeneralizedWeld.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the generalized weld.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the generalized weld is unsketched. If omitted redraw is true. If you want to unsketch several generalized welds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch generalized weld gw:

```
gw.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all generalized welds.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the generalized welds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all generalized welds in model m:

```
GeneralizedWeld.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged generalized welds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all generalized welds will be unsketched in
flag	Flag	Flag set on the generalized welds that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the generalized welds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all generalized welds flagged with flag in model m:

```
GeneralizedWeld.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

GeneralizedWeld object.

Example

To check if GeneralizedWeld property gw.example is a parameter by using the GeneralizedWeld.GetParameter() method:

```
if (gw.ViewParameters().GetParameter(gw.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for generalized weld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for generalized weld gw:

```
gw.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this generalized weld.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for generalized weld gw:

```
var xrefs = gw.Xrefs();
```

---

toString()

Description

Creates a string containing the gwld data in keyword format. Note that this contains the keyword header and the keyword cards. See also GeneralizedWeld.Keyword() and GeneralizedWeld.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for generalized weld gw in keyword format

```
var s = gw.toString();
```

Interpolation class

The Interpolation class gives you access to define *CONSTRAINED_INTERPOLATION cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AddRowData(inid*[integer]*, idof (Optional)*[integer]*, twghtx (Optional)*[real]*, twghty (Optional)*[real]*, twghtz (Optional)*[real]*, rwghtx (Optional)*[real]*, rwghty (Optional)*[real]*, rwghtz (Optional)*[real]*, cidi (Optional)*[integer]*)
- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRowData(row_index*[Integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRowData(row_index*[Integer]*)
- SetFlag(flag*[Flag]*)
- SetRowData(row_index*[Integer]*, inid*[integer]*, idof (Optional)*[integer]*, twghtx (Optional)*[real]*, twghty (Optional)*[real]*, twghtz (Optional)*[real]*, rwghtx (Optional)*[real]*, rwghty (Optional)*[real]*, rwghtz

(Optional)*[real]*, cidi (Optional)*[integer]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Interpolation constants

Name	Description
Interpolation.NODE	INID is a node.
Interpolation.NODE_SET	INID is a node set.

Interpolation properties

Name	Type	Description
cidd	integer	Coordinate System ID if LOCAL option is active.
ddof	integer	Dependent Degrees-of-Freedom.
dnid	integer	Dependent Node id.
exists (read only)	logical	true if Interpolation exists, false if referred to but not defined. (read only)
fgm	integer	Flag for special treatment of this constraint for implicit problems only.
icid	integer	Interpolation label
include	integer	The Include file number that the Interpolation is in.
indsw	integer	Switch for controlling the explicit solution when an independent (or dependent) node is deleted.
ityp	constant	The Independent Node type. Can be Interpolation.NODE or Interpolation.NODE_SET.
local	logical	true if _LOCAL is set.
model	integer	The Model number that the constrained interpolation is in.
total (read only)	integer	Total number of INID fields in the keyword.

Detailed Description

The Interpolation class allows you to create, modify, edit and manipulate *CONSTRAINED_INTERPOLATION cards. See the documentation below for more details.

Constructor

new Interpolation(Model*[Model]*, icid*[integer]*, dnid*[integer]*, inid*[integer]*, ddof (Optional)*[integer]*, local (Optional)*[boolean]*, cidd (Optional)*[integer]*, ityp (Optional)*[constant]*, idof (Optional)*[integer]*, twghtx (Optional)*[real]*, twghty (Optional)*[real]*, twghtz (Optional)*[real]*, rwghtx (Optional)*[real]*, rwghty (Optional)*[real]*, rwghtz (Optional)*[real]*, cidi (Optional)*[integer]*)

Description

Create a new Interpolation object.

Arguments

Name	Type	Description
Model	Model	Model that Interpolation will be created in
icid	integer	Interpolation label
dnid	integer	Dependent Node id.
inid	integer	Independent Node or Node Set id.
ddof (Optional)	integer	Dependent Degrees-of-Freedom. The default value is 123456.
local (Optional)	boolean	true if _LOCAL is set.
cidd (Optional)	integer	Coordinate System ID if LOCAL option is active. The default value is 0.
ityp (Optional)	constant	The Independent Node type. Can be Interpolation.NODE or Interpolation.NODE_SET. The default value is Interpolation.NODE.
idof (Optional)	integer	Independent Degrees-of-Freedom. The default value is 123456.
twghtx (Optional)	real	Weighting factor for INID. Scales the x-translational component. The default value is 1.0.
twghty (Optional)	real	Weighting factor for INID. Scales the y-translational component. The default value is twghtx.
twghtz (Optional)	real	Weighting factor for INID. Scales the z-translational component. The default value is twghtx.
rwghtx (Optional)	real	Weighting factor for INID. Scales the x-rotational component. The default value is twghtx.
rwghty (Optional)	real	Weighting factor for INID. Scales the y-rotational component. The default value is twghtx.
rwghtz (Optional)	real	Weighting factor for INID. Scales the z-rotational component. The default value is twghtx.
cidi (Optional)	integer	Coordinate System ID if LOCAL option is active. The default value is 0

Return type

Interpolation object

Example

To create a new constrained interpolation in model m, of icid 2, dnid 12, inid 10, ddof 123, local true, cidd 22, ityp NODE_SET, idof 12, and twghtx 2.24.

```
var c_i = new Interpolation(m,2,12,10,123,true,22,Interpolation.NODE_
SET,12,2.24);
```

Details of functions

AddRowData(inid*[integer]*, idof (Optional)*[integer]*, twghtx (Optional)*[real]*, twghty (Optional)*[real]*, twghtz (Optional)*[real]*, rwghtx (Optional)*[real]*, rwghty (Optional)*[real]*, rwghtz (Optional)*[real]*, cidi (Optional)*[integer]*)

Description

Used to add additional independent node card and local coordinate card (if ITYP is Interpolation.NODE_SET) to the keyword. Adds this data to the end of the selected *CONSTRAINED_INTERPOLATION

Arguments

Name	Type	Description

inid	integer	Independent Node or Node Set id.
idof (Optional)	integer	Independent Degrees-of-Freedom. The default value is 123456.
twghtx (Optional)	real	Weighting factor for INID. Scales the x-translational component. The default value is 1.0.
twghty (Optional)	real	Weighting factor for INID. Scales the y-translational component. The default value is twghtx.
twghtz (Optional)	real	Weighting factor for INID. Scales the z-translational component. The default value is twghtx.
rwghtx (Optional)	real	Weighting factor for INID. Scales the x-rotational component. The default value is twghtx.
rwghty (Optional)	real	Weighting factor for INID. Scales the y-rotational component. The default value is twghtx.
rwghtz (Optional)	real	Weighting factor for INID. Scales the z-rotational component. The default value is twghtx.
cidi (Optional)	integer	Coordinate System ID if LOCAL option is active. The default value is 0.

Return type

No return value

Example

To add INID 10 to the keyword c_i with idof 123, twghtx 1.2, twghty 2.2:

```
c_i.AddRowData(10,123,1.2,2.2);
```

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a constrained interpolation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the constrained interpolation

Return type

No return value

Example

To associate comment c to the constrained interpolation c_i:

```
c_i.AssociateComment(c);
```

Blank()

Description

Blanks the constrained interpolation

Arguments

No arguments

Return type

No return value

Example

To blank constrained interpolation c_i:

```
c_i.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained interpolations in model m:

```
Interpolation.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained interpolations will be blanked in
flag	Flag	Flag set on the constrained interpolations that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained interpolations in model m flagged with f:

```
Interpolation.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the constrained interpolation is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if constrained interpolation c_i is blanked:

```
if (c_i.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse constrained interpolation c_i:

```
c_i.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the constrained interpolation.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the constrained interpolation

Return type

No return value

Example

To clear flag f for constrained interpolation c_i:

```
c_i.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the constrained interpolation.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Interpolation object

Example

To copy constrained interpolation c_i into constrained interpolation z:

```
var z = c_i.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a Interpolation.

Arguments

Name	Type	Description
Model	Model	Model that the constrainedInterpolation will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Interpolation object (or null if not made)

Example

To start creating a constrainedInterpolation in model n:

```
var c_i = Interpolation.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a constrained interpolation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the constrained interpolation

Return type

No return value

Example

To detach comment c from the constrained interpolation c_i:

```
c_i.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit constrained interpolation c_i:

```
c_i.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for constrained interpolation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for constrained interpolation c_i:

```
c_i.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first constrained interpolation in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first constrained interpolation in

Return type

Interpolation object (or null if there are no constrained interpolations in the model).

Example

To get the first constrained interpolation in model m:

```
var c_i = Interpolation.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free constrained interpolation label in the model. Also see Interpolation.LastFreeLabel(), Interpolation.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free constrained interpolation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Interpolation label.

Example

To get the first free constrained interpolation label in model m:

```
var label = Interpolation.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the constrained interpolations in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be flagged in
flag	Flag	Flag to set on the constrained interpolations

Return type

No return value

Example

To flag all of the constrained interpolations with flag f in model m:

```
Interpolation.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the constrained interpolation is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the constrained interpolation

Return type

true if flagged, false if not.

Example

To check if constrained interpolation c_i has flag f set on it:

```
if (c_i.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each constrained interpolation in the model.
Note that ForEach has been designed to make looping over constrained interpolations as fast as possible and so has some limitations.
Firstly, a single temporary Interpolation object is created and on each function call it is updated with the current constrained interpolation data. This means that you should not try to store the Interpolation object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new constrained interpolations inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations are in
func	function	Function to call for each constrained interpolation
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the constrained interpolations in model m:

```
Interpolation.ForEach(m, test);
function test(c_i)
{
// c_i is Interpolation object
}
```

To call function test for all of the constrained interpolations in model m with optional object:

```
var data = { x:0, y:0 };
Interpolation.ForEach(m, test, data);
function test(c_i, extra)
{
// c_i is Interpolation object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Interpolation objects for all of the constrained interpolations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained interpolations from

Return type

Array of Interpolation objects

Example

To make an array of Interpolation objects for all of the constrained interpolations in model m

```
var c_i = Interpolation.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a constrained interpolation.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the constrained interpolation c_i:

```
var comm_array = c_i.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Interpolation objects for all of the flagged constrained interpolations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained interpolations from
flag	Flag	Flag set on the constrained interpolations that you want to retrieve

Return type

Array of Interpolation objects

Example

To make an array of Interpolation objects for all of the constrained interpolations in model m flagged with f

```
var c_i = Interpolation.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Interpolation object for a constrained interpolation ID.

Arguments

Name	Type	Description
Model	Model	Model to find the constrained interpolation in
number	integer	number of the constrained interpolation you want the Interpolation object for

Return type

Interpolation object (or null if constrained interpolation does not exist).

Example

To get the Interpolation object for constrained interpolation 100 in model m

```
var c_i = Interpolation.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Interpolation property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Interpolation.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	constrained interpolation property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Interpolation property c_i.example is a parameter:

```
Options.property_parameter_names = true;
if (c_i.GetParameter(c_i.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Interpolation property c_i.example is a parameter by using the GetParameter method:

```
if (c_i.ViewParameters().GetParameter(c_i.example) ) do_something...
```

GetRowData(row_index*[Integer]*)

Description

Returns independent node cards and local coordinate cards (if ITYP is Interpolation.NODE_SET) for the selected row of the *CONSTRAINED_INTERPOLATION.

Arguments

Name	Type	Description
row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Interpolation.total

Return type

Array containing data.

Example

To loop over all the lines of the keyword for c_i:

```
for (i=0; i<c_i.total; i++)
        var data = c_i.GetRowData(i);
```

Keyword()

Description

Returns the keyword for this Interpolation (*constrained_interpolation). **Note that a carriage return is not added**. See also Interpolation.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Interpolation c_i:

```
var key = c_i.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Interpolation. **Note that a carriage return is not added**. See also Interpolation.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Interpolation c_i:

```
var cards = c_i.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last constrained interpolation in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last constrained interpolation in

Return type

Interpolation object (or null if there are no constrained interpolations in the model).

Example

To get the last constrained interpolation in model m:

```
var c_i = Interpolation.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free constrained interpolation label in the model. Also see Interpolation.FirstFreeLabel(), Interpolation.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free constrained interpolation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Interpolation label.

Example

To get the last free constrained interpolation label in model m:

```
var label = Interpolation.LastFreeLabel(m);
```

Next()

Description

Returns the next constrained interpolation in the model.

Arguments

No arguments

Return type

Interpolation object (or null if there are no more constrained interpolations in the model).

Example

To get the constrained interpolation in model m after constrained interpolation c_i:

```
var c_i = c_i.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) constrained interpolation label in the model. Also see Interpolation.FirstFreeLabel(), Interpolation.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free constrained interpolation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Interpolation label.

Example

To get the next free constrained interpolation label in model m:

```
var label = Interpolation.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a constrained interpolation.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained interpolations from that model can be picked. If the argument is a Flag then only constrained interpolations that are flagged with *limit* can be selected. If omitted, or null, any constrained interpolations from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Interpolation object (or null if not picked)

Example

To pick a constrained interpolation from model m giving the prompt 'Pick constrained interpolation from screen':

```
var c_i = Interpolation.Pick('Pick constrained interpolation from screen', m);
```

Previous()

Description

Returns the previous constrained interpolation in the model.

Arguments

No arguments

Return type

Interpolation object (or null if there are no more constrained interpolations in the model).

Example

To get the constrained interpolation in model m before constrained interpolation c_i:

```
var c_i = c_i.Previous();
```

RemoveRowData(row_index*[Integer]*)

Description

Removes an independent node card and a local coordinate card (if ITYP is Interpolation.NODE_SET) for the selected row on the *CONSTRAINED_INTERPOLATION.

Arguments

Name	**Type**	**Description**
row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Interpolation.total

Return type

No return value.

Example

To remove row 2 for c_i:

```
c_i.RemoveRowData(1);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the constrained interpolations in model m, from 1000000:

```
Interpolation.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained interpolations will be renumbered in
flag	Flag	Flag set on the constrained interpolations that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the constrained interpolations in model m flagged with f, from 1000000:

```
Interpolation.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select constrained interpolations using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting constrained interpolations
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained interpolations from that model can be selected. If the argument is a Flag then only constrained interpolations that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any constrained interpolations can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of constrained interpolations selected or null if menu cancelled

Example

To select constrained interpolations from model m, flagging those selected with flag f, giving the prompt 'Select constrained interpolations':

```
Interpolation.Select(f, 'Select constrained interpolations', m);
```

To select constrained interpolations, flagging those selected with flag f but limiting selection to constrained interpolations flagged with flag l, giving the prompt 'Select constrained interpolations':

```
Interpolation.Select(f, 'Select constrained interpolations', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the constrained interpolation.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the constrained interpolation

Return type

No return value

Example

To set flag f for constrained interpolation c_i:

```
c_i.SetFlag(f);
```

SetRowData(row_index*[Integer]*, inid*[integer]*, idof (Optional)*[integer]*, twghtx (Optional)*[real]*, twghty (Optional)*[real]*, twghtz (Optional)*[real]*, rwghtx (Optional)*[real]*, rwghty (Optional)*[real]*, rwghtz (Optional)*[real]*, cidi (Optional)*[integer]*)

Description

Used to reset values in already existing independent node cards and local coordinate cards (if ITYP is Interpolation.NODE_SET) in the selected row of *CONSTRAINED_INTERPOLATION

Arguments

Name	Type	Description

row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Interpolation.total
inid	integer	Independent Node or Node Set id.
idof (Optional)	integer	Independent Degrees-of-Freedom. The default value is 123456.
twghtx (Optional)	real	Weighting factor for INID. Scales the x-translational component. The default value is 1.0.
twghty (Optional)	real	Weighting factor for INID. Scales the y-translational component. The default value is twghtx.
twghtz (Optional)	real	Weighting factor for INID. Scales the z-translational component. The default value is twghtx.
rwghtx (Optional)	real	Weighting factor for INID. Scales the x-rotational component. The default value is twghtx.
rwghty (Optional)	real	Weighting factor for INID. Scales the y-rotational component. The default value is twghtx.
rwghtz (Optional)	real	Weighting factor for INID. Scales the z-rotational component. The default value is twghtx.
cidi (Optional)	integer	Coordinate System ID if LOCAL option is active. The default value is 0

Return type

No return value

Example

To reset the values of row 3 of the keyword with INID 11, idof 1234, twghtx 2.2, twghty 4.2:

```
c_i.SetRowData(2,11,1234,2.2,4.2);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the constrained interpolation. The constrained interpolation will be sketched until you either call Interpolation.Unsketch(), Interpolation.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained interpolation is sketched. If omitted redraw is true. If you want to sketch several constrained interpolations and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch constrained interpolation c_i:

```
c_i.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged constrained interpolations in the model. The constrained interpolations will be sketched until you either call Interpolation.Unsketch(), Interpolation.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained interpolations will be sketched in
flag	Flag	Flag set on the constrained interpolations that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the constrained interpolations are sketched. If omitted redraw is true. If you want to sketch flagged constrained interpolations several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all constrained interpolations flagged with flag in model m:

```
Interpolation.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing constrained interpolations should be counted. If false or omitted referenced but undefined constrained interpolations will also be included in the total.

Return type

number of constrained interpolations

Example

To get the total number of constrained interpolations in model m:

```
var total = Interpolation.Total(m);
```

Unblank()

Description

Unblanks the constrained interpolation

Arguments

No arguments

Return type

No return value

Example

To unblank constrained interpolation c_i:

```
c_i.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained interpolations in model m:

```
Interpolation.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged constrained interpolations will be unblanked in
flag	Flag	Flag set on the constrained interpolations that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained interpolations in model m flagged with f:

```
Interpolation.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all constrained interpolations will be unset in

flag	Flag	Flag to unset on the constrained interpolations

Return type

No return value

Example

To unset the flag f on all the constrained interpolations in model m:

```
Interpolation.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the constrained interpolation.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained interpolation is unsketched. If omitted redraw is true. If you want to unsketch several constrained interpolations and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch constrained interpolation c_i:

```
c_i.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all constrained interpolations.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the constrained interpolations are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained interpolations in model m:

```
Interpolation.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged constrained interpolations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained interpolations will be unsketched in
flag	Flag	Flag set on the constrained interpolations that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the constrained interpolations are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained interpolations flagged with flag in model m:

```
Interpolation.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Interpolation object.

Example

To check if Interpolation property c_i.example is a parameter by using the Interpolation.GetParameter() method:

```
if (c_i.ViewParameters().GetParameter(c_i.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for constrained interpolation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give

details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for constrained interpolation c_i:

```
c_i.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this constrained interpolation.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for constrained interpolation c_i:

```
var xrefs = c_i.Xrefs();
```

---

toString()

Description

Creates a string containing the Interpolation data in keyword format. Note that this contains the keyword header and the keyword cards. See also Interpolation.Keyword() and Interpolation.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Interpolation c_i in keyword format

```
var s = c_i.toString();
```

InterpolationSpotweld (Spr3) class

The InterpolationSpotweld class gives you access to constrained Interpolation Spotweld (spr3) cards in PRIMER.
More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

InterpolationSpotweld constants

Constants for Flags for Interpolation

Name	Description
InterpolationSpotweld.INVERSE	Property INTP value EQ.2.0: Inverse distance weighting.
InterpolationSpotweld.LINEAR	Property INTP value EQ.0.0: Linear (default).
InterpolationSpotweld.UNIFORM	Property INTP value EQ.1.0: Uniform.

Constants for Model Values

Name	Description
InterpolationSpotweld.SPR3	Property MODEL value EQ.1.0: SPR3 (default).
InterpolationSpotweld.SPR3_MAT_PARAM	Property MODEL value EQ.11.0: Same as 1 with selected material parameters as functions.
InterpolationSpotweld.SPR3_MAT_PARAM_MOD	Property MODEL value EQ.21.0: Same as 11 with slight modification.
InterpolationSpotweld.SPR4	Property MODEL value EQ.2.0: SPR4.
InterpolationSpotweld.SPR4_MAT_PARAM	Property MODEL value EQ.12.0: Same as 2 with selected material parameters as functions.
InterpolationSpotweld.SPR4_MAT_PARAM_MOD	Property MODEL value EQ.22.0: Same as 12 with slight modification.

InterpolationSpotweld properties

Name	Type	Description
alpha1	real/integer	Scaling factor alpha 1. Function ID if MODEL > 10.
alpha2	real	Plastic initiation displacement scaling factor alpha2.
alpha3	real	Plastic initiation displacement scaling factor alpha3.
beta	real	Exponent for plastic potential beta 1. Function ID if MODEL > 10.
beta2	real	Exponent for plastic initiation displacement beta2.
beta3	real	Exponent for plastic initiation displacement beta3.
dens	real	Spotweld density (necessary for time step calculation).
exists	logical	true if constrained interpolation spotweld exists, false if referred to but not defined. (read only)
gamma	real	Scaling factor.
include	integer	The Include file number that the constrained interpolation spotweld is in.
intp	real	Flag for interpolation. Values can be InterpolationSpotweld.LINEAR, InterpolationSpotweld.UNIFORM or InterpolationSpotweld.INVERSE.
lcdexp	integer	Load curve ID for damage exponent vs. mode mixity
lcf	integer	Load curve ID describing force versus plastic displacement.
lcupf	integer	Load curve ID describing plastic initiation displacement versus mode mixity. Required only for MODEL values = InterpolationSpotweld.SPR3, InterpolationSpotweld.SPR3_MAT_PARAM or InterpolationSpotweld.SPR3_MAT_PARAM_MOD.
lcupr	integer	Load curve ID describing plastic rupture displacement versus mode mixity. Required only for MODEL values = InterpolationSpotweld.SPR3, InterpolationSpotweld.SPR3_MAT_PARAM or InterpolationSpotweld.SPR3_MAT_PARAM_MOD.
model	integer	The Model number that the interpolation spotweld is in.
mrn	real	Proportionality factor for dependency RN.

mrs	real	Proportionality factor for dependency RS.
nsid	integer	Node Set ID of spotweld location nodes.
pid1	integer	Part ID of first sheet.
pid2	integer	Part ID
pidvb	real	Part ID for visualization beams representing SPR3 in post-processing.
r	real	Spotweld Radius.
rn	real/integer	Tensile strength factor. Function ID if MODEL > 10.
rs	real	Shear strength factor. Function ID if MODEL > 10.
sropt	real	Shear rotation option.
stf	real/integer	Elastic stiffness OR material ID if less than 0. Function ID if MODEL > 10.
stiff2	real	Elastic shear stiffness.
stiff3	real	Elastic bending stiffness.
stiff4	real	Elastic torsional stiffness.
thick	real	Total thickness of both sheets.
upfn	real	Plastic initiation displacement in normal direction.
upfs	real	Plastic initiation displacement in shear direction.
uprn	real	Plastic rupture displacement in normal direction.
uprs	real	Plastic rupture displacement in shear direction.

Detailed Description

The InterpolationSpotweld class allows you to create, modify, edit and manipulate constrained interpolation spotweld (spr3) cards. See the documentation below for more details.

For convenience "Spr3" can also be used as the class name instead of "InterpolationSpotweld".

Constructor

new InterpolationSpotweld(Model*[Model]*, pid1*[integer]*, pid2*[integer]*, nsid*[integer]*)

Description

Create a new InterpolationSpotweld object.

Arguments

Name	Type	Description
Model	Model	Model that constrained interpolation spotweld will be created in
pid1	integer	Part ID of first sheet.
pid2	integer	Part ID of second sheet.
nsid	integer	Node Set ID of spotweld location nodes.

Return type

InterpolationSpotweld object

Example

To create a new constrained interpolation spotweld in model m with first sheet 100, second sheet 200 and spotweld node set 100

```
var s = new InterpolationSpotweld(m, 100, 200, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a interpolation spotweld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the interpolation spotweld

Return type

No return value

Example

To associate comment c to the interpolation spotweld s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the interpolation spotweld

Arguments

No arguments

Return type

No return value

Example

To blank interpolation spotweld s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all interpolation spotwelds will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the interpolation spotwelds in model m:

```
InterpolationSpotweld.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged interpolation spotwelds will be blanked in
flag	Flag	Flag set on the interpolation spotwelds that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the interpolation spotwelds in model m flagged with f:

```
InterpolationSpotweld.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the interpolation spotweld is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if interpolation spotweld s is blanked:

```
if (s.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the interpolation spotweld.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the interpolation spotweld

Return type

No return value

Example

To clear flag f for interpolation spotweld s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the interpolation spotweld.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

InterpolationSpotweld object

Example

To copy interpolation spotweld s into interpolation spotweld z:

```
var z = s.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a interpolation spotweld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the interpolation spotweld

Return type

No return value

Example

To detach comment c from the interpolation spotweld s:

```
s.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for interpolation spotweld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for interpolation spotweld s:

```
s.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first interpolation spotweld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first interpolation spotweld in

Return type

InterpolationSpotweld object (or null if there are no interpolation spotwelds in the model).

Example

To get the first interpolation spotweld in model m:

```
var s = InterpolationSpotweld.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the interpolation spotwelds in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all interpolation spotwelds will be flagged in

flag	Flag	Flag to set on the interpolation spotwelds

Return type

No return value

Example

To flag all of the interpolation spotwelds with flag f in model m:

```
InterpolationSpotweld.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the interpolation spotweld is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the interpolation spotweld

Return type

true if flagged, false if not.

Example

To check if interpolation spotweld s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each interpolation spotweld in the model.
Note that ForEach has been designed to make looping over interpolation spotwelds as fast as possible and so has some limitations.
Firstly, a single temporary InterpolationSpotweld object is created and on each function call it is updated with the current interpolation spotweld data. This means that you should not try to store the InterpolationSpotweld object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new interpolation spotwelds inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all interpolation spotwelds are in
func	function	Function to call for each interpolation spotweld
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the interpolation spotwelds in model m:

```
InterpolationSpotweld.ForEach(m, test);
function test(s)
{
// s is InterpolationSpotweld object
}
```

To call function test for all of the interpolation spotwelds in model m with optional object:

```
var data = { x:0, y:0 };
InterpolationSpotweld.ForEach(m, test, data);
function test(s, extra)
{
// s is InterpolationSpotweld object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of InterpolationSpotweld objects for all of the interpolation spotwelds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interpolation spotwelds from

Return type

Array of InterpolationSpotweld objects

Example

To make an array of InterpolationSpotweld objects for all of the interpolation spotwelds in model m

```
var s = InterpolationSpotweld.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a interpolation spotweld.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the interpolation spotweld s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of InterpolationSpotweld objects for all of the flagged interpolation spotwelds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interpolation spotwelds from
flag	Flag	Flag set on the interpolation spotwelds that you want to retrieve

Return type

Array of InterpolationSpotweld objects

Example

To make an array of InterpolationSpotweld objects for all of the interpolation spotwelds in model m flagged with f

```
var s = InterpolationSpotweld.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the InterpolationSpotweld object for a interpolation spotweld ID.

Arguments

Name	Type	Description
Model	Model	Model to find the interpolation spotweld in
number	integer	number of the interpolation spotweld you want the InterpolationSpotweld object for

Return type

InterpolationSpotweld object (or null if interpolation spotweld does not exist).

Example

To get the InterpolationSpotweld object for interpolation spotweld 100 in model m

```
var s = InterpolationSpotweld.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a InterpolationSpotweld property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the InterpolationSpotweld.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	interpolation spotweld property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if InterpolationSpotweld property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if InterpolationSpotweld property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this interpolation spotweld (*CONSTRAINED_INTERPOLATION_SPOTWELD). **Note that a carriage return is not added**. See also InterpolationSpotweld.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for interpolation spotweld s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the interpolation spotweld. **Note that a carriage return is not added**. See also InterpolationSpotweld.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for interpolation spotweld s:

```
var cards = s.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last interpolation spotweld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last interpolation spotweld in

Return type

InterpolationSpotweld object (or null if there are no interpolation spotwelds in the model).

Example

To get the last interpolation spotweld in model m:

```
var s = InterpolationSpotweld.Last(m);
```

Next()

Description

Returns the next interpolation spotweld in the model.

Arguments

No arguments

Return type

InterpolationSpotweld object (or null if there are no more interpolation spotwelds in the model).

Example

To get the interpolation spotweld in model m after interpolation spotweld s:

```
var s = s.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a interpolation spotweld.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only interpolation spotwelds from that model can be picked. If the argument is a Flag then only interpolation spotwelds that are flagged with *limit* can be selected. If omitted, or null, any interpolation spotwelds from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

InterpolationSpotweld object (or null if not picked)

Example

To pick a interpolation spotweld from model m giving the prompt 'Pick interpolation spotweld from screen':

```
var s = InterpolationSpotweld.Pick('Pick interpolation spotweld from screen',
m);
```

Previous()

Description

Returns the previous interpolation spotweld in the model.

Arguments

No arguments

Return type

InterpolationSpotweld object (or null if there are no more interpolation spotwelds in the model).

Example

To get the interpolation spotweld in model m before interpolation spotweld s:

```
var s = s.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select interpolation spotwelds using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting interpolation spotwelds
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only interpolation spotwelds from that model can be selected. If the argument is a Flag then only interpolation spotwelds that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any interpolation spotwelds can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of interpolation spotwelds selected or null if menu cancelled

Example

To select interpolation spotwelds from model m, flagging those selected with flag f, giving the prompt 'Select interpolation spotwelds':

```
InterpolationSpotweld.Select(f, 'Select interpolation spotwelds', m);
```

To select interpolation spotwelds, flagging those selected with flag f but limiting selection to interpolation spotwelds flagged with flag l, giving the prompt 'Select interpolation spotwelds':

```
InterpolationSpotweld.Select(f, 'Select interpolation spotwelds', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the interpolation spotweld.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the interpolation spotweld

Return type

No return value

Example

To set flag f for interpolation spotweld s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the interpolation spotweld. The interpolation spotweld will be sketched until you either call InterpolationSpotweld.Unsketch(), InterpolationSpotweld.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the interpolation spotweld is sketched. If omitted redraw is true. If you want to sketch several interpolation spotwelds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch interpolation spotweld s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged interpolation spotwelds in the model. The interpolation spotwelds will be sketched until you either call InterpolationSpotweld.Unsketch(), InterpolationSpotweld.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged interpolation spotwelds will be sketched in
flag	Flag	Flag set on the interpolation spotwelds that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the interpolation spotwelds are sketched. If omitted redraw is true. If you want to sketch flagged interpolation spotwelds several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all interpolation spotwelds flagged with flag in model m:

```
InterpolationSpotweld.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing interpolation spotwelds should be counted. If false or omitted referenced but undefined interpolation spotwelds will also be included in the total.

Return type

number of interpolation spotwelds

Example

To get the total number of interpolation spotwelds in model m:

```
var total = InterpolationSpotweld.Total(m);
```

Unblank()

Description

Unblanks the interpolation spotweld

Arguments

No arguments

Return type

No return value

Example

To unblank interpolation spotweld s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the interpolation spotwelds in the model.

Arguments

Name	Type	Description

Model	Model	Model that all interpolation spotwelds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the interpolation spotwelds in model m:

```
InterpolationSpotweld.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged interpolation spotwelds will be unblanked in
flag	Flag	Flag set on the interpolation spotwelds that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the interpolation spotwelds in model m flagged with f:

```
InterpolationSpotweld.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all interpolation spotwelds will be unset in
flag	Flag	Flag to unset on the interpolation spotwelds

Return type

No return value

Example

To unset the flag f on all the interpolation spotwelds in model m:

```
InterpolationSpotweld.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the interpolation spotweld.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the interpolation spotweld is unsketched. If omitted redraw is true. If you want to unsketch several interpolation spotwelds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch interpolation spotweld s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all interpolation spotwelds.

Arguments

Name	Type	Description
Model	Model	Model that all interpolation spotwelds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the interpolation spotwelds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all interpolation spotwelds in model m:

```
InterpolationSpotweld.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged interpolation spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all interpolation spotwelds will be unsketched in
flag	Flag	Flag set on the interpolation spotwelds that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the interpolation spotwelds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all interpolation spotwelds flagged with flag in model m:

```
InterpolationSpotweld.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

InterpolationSpotweld object.

Example

To check if InterpolationSpotweld property s.example is a parameter by using the InterpolationSpotweld.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for interpolation spotweld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for interpolation spotweld s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this interpolation spotweld.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for interpolation spotweld s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the interpolation spotweld data in keyword format. Note that this contains the keyword header and the keyword cards. See also InterpolationSpotweld.Keyword() and InterpolationSpotweld.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for interpolation spotweld s in keyword format

```
var str = s.toString();
```

Joint class

The Joint class gives you access to constrained joint cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Joint constants

Name	Description
Joint.CONSTANT_VELOCITY	CONSTANT_VELOCITY is *CONSTRAINED_JOINT_CONSTANT_VELOCITY.
Joint.CYLINDRICAL	CYLINDRICAL is *CONSTRAINED_JOINT_CYLINDRICAL.
Joint.GEARS	GEARS is *CONSTRAINED_JOINT_GEARS.
Joint.LOCKING	LOCKING is *CONSTRAINED_JOINT_LOCKING.
Joint.PLANAR	PLANAR is *CONSTRAINED_JOINT_PLANAR.
Joint.PULLEY	PULLEY is *CONSTRAINED_JOINT_PULLEY.
Joint.RACK_AND_PINION	RACK_AND_PINION is *CONSTRAINED_JOINT_RACK_AND_PINION.
Joint.REVOLUTE	REVOLUTE is *CONSTRAINED_JOINT_REVOLUTE.
Joint.ROTATIONAL_MOTOR	ROTATIONAL_MOTOR is *CONSTRAINED_JOINT_ROTATIONAL_MOTOR.
Joint.SCREW	SCREW is *CONSTRAINED_JOINT_SCREW.
Joint.SPHERICAL	SPHERICAL is *CONSTRAINED_JOINT_SPHERICAL.
Joint.TRANSLATIONAL	TRANSLATIONAL is *CONSTRAINED_JOINT_TRANSLATIONAL.
Joint.TRANSLATIONAL_MOTOR	TRANSLATIONAL_MOTOR is *CONSTRAINED_JOINT_TRANSLATIONAL_MOTOR.
Joint.UNIVERSAL	UNIVERSAL is *CONSTRAINED_JOINT_UNIVERSAL.

Joint properties

Name	Type	Description
cid	integer	Coordinate system number.
coupl	real	Coupling between force and moment failure.
damp	real	Damping scale factor.
exists	logical	true if constrained joint exists, false if referred to but not defined. (read only)
failure	logical	true if _FAILURE option is set, false if not.
h_angle	real	Helix angle for gears.
heading	string	Constrained joint heading.
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the constrained joint is in.
jid	integer	Constrained joint number (identical to label).
label	integer	Constrained joint number.
lcid	integer	Loadcuve number.
local	logical	true if _LOCAL option is set, false if not.
lst	integer	Local system type is accelerometer if lst is 1, rigid body if 0.
model	integer	The Model number that the joint is in.
mxx	real	Torsional moment resultant at failure.
myy	real	Moment resultant at failure.

mzz	real	Moment resultant at failure.
n1	integer	Node number 1.
n2	integer	Node number 2.
n3	integer	Node number 3.
n4	integer	Node number 4.
n5	integer	Node number 5.
n6	integer	Node number 6.
nxx	real	Axial force resultant at failure.
nyy	real	Force resultant at failure.
nzz	real	Force resultant at failure.
option	constant	The Constrained Joint option. Can be: Joint.SPHERICAL, Joint.REVOLUTE, Joint.CYLINDRICAL, Joint.PLANAR, Joint.UNIVERSAL, Joint.TRANSLATIONAL, Joint.LOCKING, Joint.TRANSLATIONAL_MOTOR, Joint.ROTATIONAL_MOTOR, Joint.GEARS, Joint.RACK_AND_PINION, Joint.CONSTANT_VELOCITY, Joint.PULLEY or Joint.SCREW
parm	real	Parameter for function.
r1	real	Gear and pulley radius.
raid	integer	Rigid body or accelerometer number.
rps	real	Relative penalty stiffness.
tfail	real	Time for joint failure.
type	integer	Flag for motor type.

Detailed Description

The Joint class allows you to create, modify, edit and manipulate constrained joint cards. See the documentation below for more details.

Constructor

new Joint(Model*[Model]*, option*[constant]*, n1*[integer]*, n2*[integer]*, jid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Joint object.

Arguments

Name	Type	Description
Model	Model	Model that constrained joint will be created in
option	constant	Constrained joint type (any).
n1	integer	Node 1.
n2	integer	Node 2.
jid (optional)	integer	Constrained joint number.
heading (optional)	string	Constrained joint title.

Return type

Joint object

Example

To create a new constrained joint 500 called "test spherical joint" of type _SPHERICAL in model m with nodes 50 and 150

```
var j = new Joint(m, Joint.SPHERICAL, 50, 150, 500, "test spherical joint");
```

To create a new constrained joint 500 called "test revolute joint" of type _REVOLUTE in model m with nodes 50, 100, 150 and 200

```
var j = new Joint(m, Joint.REVOLUTE, 50, 100, 500, "test revolute joint");
j.n3 = 150;
j.n4 = 200;
```

new Joint(Model*[Model]*, option*[constant]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, jid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Joint object.

Arguments

Name	Type	Description
Model	Model	Model that constrained joint will be created in
option	constant	Constrained joint type. Can be Joint.REVOLUTE, Joint.CYLINDRICAL, Joint.PLANAR, Joint.UNIVERSAL or Joint.TRANSLATIONAL_MOTOR.
n1	integer	Node 1.
n2	integer	Node 2.
n3	integer	Node 3.
n4	integer	Node 4.
jid (optional)	integer	Constrained joint number.
heading (optional)	string	Constrained joint title.

Return type

Joint object

Example

To create a new constrained joint 500 called "test revolute joint" of type _REVOLUTE in model m with nodes 50, 100, 150 and 200

```
var j = new Joint(m, Joint.REVOLUTE, 50, 100, 150, 200, 500, "test revolute
joint");
```

new Joint(Model*[Model]*, option*[constant]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, n5*[integer]*, n6*[integer]*, jid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Joint object.

Arguments

Name	Type	Description
Model	Model	Model that constrained joint will be created in

option	constant	Constrained joint type. Can be Joint.TRANSLATIONAL, Joint.LOCKING, Joint.ROTATIONAL_MOTOR, Joint.GEARS, Joint.RACK_AND_PINION, Joint.CONSTANT_VELOCITY, Joint.PULLEY or Joint.SCREW.
n1	integer	Node 1.
n2	integer	Node 2.
n3	integer	Node 3.
n4	integer	Node 4.
n5	integer	Node 5.
n6	integer	Node 6.
jid (optional)	integer	Constrained joint number.
heading (optional)	string	Constrained joint title.

Return type

Joint object

Example

To create a new constrained joint 500 called "test translational joint" of type _TRANSLATIONAL in model m with nodes 50, 100, 150, 300, 250 and 300

```
var j = new Joint(m, Joint.TRANSLATIONAL, 50, 100, 150, 200, 250, 300, 500,
"test translational joint");
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a joint.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the joint

Return type

No return value

Example

To associate comment c to the joint j:

```
j.AssociateComment(c);
```

Blank()

Description

Blanks the joint

Arguments

No arguments

Return type

No return value

Example

To blank joint j:

```
j.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the joints in model m:

```
Joint.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joints will be blanked in
flag	Flag	Flag set on the joints that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the joints in model m flagged with f:

```
Joint.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the joint is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if joint j is blanked:

```
if (j.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse joint j:

```
j.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the joint.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the joint

Return type

No return value

Example

To clear flag f for joint j:

```
j.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the joint.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Joint object

Example

To copy joint j into joint z:

```
var z = j.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a joint.

Arguments

Name	Type	Description
Model	Model	Model that the joint will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Joint object (or null if not made)

Example

To start creating a joint in model n:

```
var j = Joint.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a joint.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the joint

Return type

No return value

Example

To detach comment c from the joint j:

```
j.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit joint j:

```
j.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for joint. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for joint j:

```
j.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first joint in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first joint in

Return type

Joint object (or null if there are no joints in the model).

Example

To get the first joint in model m:

```
var j = Joint.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free joint label in the model. Also see Joint.LastFreeLabel(), Joint.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free joint label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Joint label.

Example

To get the first free joint label in model m:

```
var label = Joint.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the joints in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be flagged in
flag	Flag	Flag to set on the joints

Return type

No return value

Example

To flag all of the joints with flag f in model m:

```
Joint.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the joint is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the joint

Return type

true if flagged, false if not.

Example

To check if joint j has flag f set on it:

```
if (j.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each joint in the model.
Note that ForEach has been designed to make looping over joints as fast as possible and so has some limitations. Firstly, a single temporary Joint object is created and on each function call it is updated with the current joint data. This means that you should not try to store the Joint object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new joints inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all joints are in
func	function	Function to call for each joint
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the joints in model m:

```
Joint.ForEach(m, test);
function test(j)
{
// j is Joint object
}
```

To call function test for all of the joints in model m with optional object:

```
var data = { x:0, y:0 };
Joint.ForEach(m, test, data);
function test(j, extra)
{
// j is Joint object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Joint objects for all of the joints in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get joints from

Return type

Array of Joint objects

Example

To make an array of Joint objects for all of the joints in model m

```
var j = Joint.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a joint.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the joint j:

```
var comm_array = j.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of Joint objects for all of the flagged joints in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get joints from
flag	Flag	Flag set on the joints that you want to retrieve

Return type

Array of Joint objects

Example

To make an array of Joint objects for all of the joints in model m flagged with f

```
var j = Joint.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Joint object for a joint ID.

Arguments

Name	Type	Description
Model	Model	Model to find the joint in
number	integer	number of the joint you want the Joint object for

Return type

Joint object (or null if joint does not exist).

Example

To get the Joint object for joint 100 in model m

```
var j = Joint.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Joint property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Joint.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	joint property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Joint property j.example is a parameter:

```
Options.property_parameter_names = true;
if (j.GetParameter(j.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Joint property j.example is a parameter by using the GetParameter method:

```
if (j.ViewParameters().GetParameter(j.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this joint (*CONSTRAINED_JOINT). **Note that a carriage return is not added**. See also Joint.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for joint j:

```
var key = j.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the joint. **Note that a carriage return is not added**. See also Joint.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for joint j:

```
var cards = j.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last joint in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last joint in

Return type

Joint object (or null if there are no joints in the model).

Example

To get the last joint in model m:

```
var j = Joint.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free joint label in the model. Also see Joint.FirstFreeLabel(), Joint.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free joint label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Joint label.

Example

To get the last free joint label in model m:

```
var label = Joint.LastFreeLabel(m);
```

Next()

Description

Returns the next joint in the model.

Arguments

No arguments

Return type

Joint object (or null if there are no more joints in the model).

Example

To get the joint in model m after joint j:

```
var j = j.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) joint label in the model. Also see Joint.FirstFreeLabel(), Joint.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free joint label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Joint label.

Example

To get the next free joint label in model m:

```
var label = Joint.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a joint.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only joints from that model can be picked. If the argument is a Flag then only joints that are flagged with *limit* can be selected. If omitted, or null, any joints from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Joint object (or null if not picked)

Example

To pick a joint from model m giving the prompt 'Pick joint from screen':

```
var j = Joint.Pick('Pick joint from screen', m);
```

Previous()

Description

Returns the previous joint in the model.

Arguments

No arguments

Return type

Joint object (or null if there are no more joints in the model).

Example

To get the joint in model m before joint j:

```
var j = j.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be renumbered in

start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the joints in model m, from 1000000:

```
Joint.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joints will be renumbered in
flag	Flag	Flag set on the joints that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the joints in model m flagged with f, from 1000000:

```
Joint.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select joints using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting joints
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only joints from that model can be selected. If the argument is a Flag then only joints that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any joints can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of joints selected or null if menu cancelled

Example

To select joints from model m, flagging those selected with flag f, giving the prompt 'Select joints':

```
Joint.Select(f, 'Select joints', m);
```

To select joints, flagging those selected with flag f but limiting selection to joints flagged with flag l, giving the prompt 'Select joints':

```
Joint.Select(f, 'Select joints', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the joint.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the joint

Return type

No return value

Example

To set flag f for joint j:

```
j.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the joint. The joint will be sketched until you either call Joint.Unsketch(), Joint.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the joint is sketched. If omitted redraw is true. If you want to sketch several joints and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch joint j:

```
j.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged joints in the model. The joints will be sketched until you either call Joint.Unsketch(), Joint.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joints will be sketched in
flag	Flag	Flag set on the joints that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the joints are sketched. If omitted redraw is true. If you want to sketch flagged joints several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all joints flagged with flag in model m:

```
Joint.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of joints in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing joints should be counted. If false or omitted referenced but undefined joints will also be included in the total.

Return type

number of joints

Example

To get the total number of joints in model m:

```
var total = Joint.Total(m);
```

Unblank()

Description

Unblanks the joint

Arguments

No arguments

Return type

No return value

Example

To unblank joint j:

```
j.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the joints in model m:

```
Joint.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged joints will be unblanked in
flag	Flag	Flag set on the joints that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the joints in model m flagged with f:

```
Joint.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all joints will be unset in

flag	Flag	Flag to unset on the joints

Return type

No return value

Example

To unset the flag f on all the joints in model m:

```
Joint.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the joint.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the joint is unsketched. If omitted redraw is true. If you want to unsketch several joints and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch joint j:

```
j.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all joints.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the joints are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all joints in model m:

```
Joint.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged joints in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joints will be unsketched in
flag	Flag	Flag set on the joints that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the joints are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all joints flagged with flag in model m:

```
Joint.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Joint object.

Example

To check if Joint property j.example is a parameter by using the Joint.GetParameter() method:

```
if (j.ViewParameters().GetParameter(j.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for joint. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for joint j:

```
j.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this joint.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for joint j:

```
var xrefs = j.Xrefs();
```

---

toString()

Description

Creates a string containing the joint data in keyword format. Note that this contains the keyword header and the keyword cards. See also Joint.Keyword() and Joint.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for joint j in keyword format

```
var s = j.toString();
```

JointStiffness (Jstf) class

The JointStiffness class gives you access to constrained joint stiffness cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

JointStiffness constants

Name	Description
JointStiffness.CYLINDRICAL	JointStiffness is *CONSTRAINED_JOINT_STIFFNESS_CYLINDRICAL.
JointStiffness.FLEXION_ TORSION	JointStiffness is *CONSTRAINED_JOINT_STIFFNESS_ FLEXION-TORSION.
JointStiffness.GENERALIZED	JointStiffness is *CONSTRAINED_JOINT_STIFFNESS_GENERALIZED.
JointStiffness.TRANSLATIONAL	JointStiffness is *CONSTRAINED_JOINT_STIFFNESS_TRANSLATIONAL.

JointStiffness properties

Name	Type	Description
cida	integer	Coordinate System ID #A.
cidb	integer	Coordinate System ID #B.
dlcidal	integer	LC: Alpha damping moment vs Rotl vel.
dlcidbt	integer	LC: Beta damping moment vs Rotl vel.
dlcidg	integer	LC: Gamma damping factor vs factor on Alpha damping moment.
dlcidp	integer	LC: P damping vs P rel velocity.
dlcidph	integer	LC: Phi damping moment vs rotation vel.
dlcidps	integer	LC: Psi damping moment vs rotation vel.
dlcidr	integer	LC: R damping vs R rel velocity.
dlcidt	integer	LC: Theta damping moment vs rotation vel.
dlcidx	integer	LC: X damping vs X rel velocity.
dlcidy	integer	LC: Y damping vs Y rel velocity.
dlcidz	integer	LC: Z damping vs Z rel velocity.
esal	real	Stiffness/angle in Alpha direction.
esbt	real	Stiffness/angle in Beta direction.
esph	real	Stiffness/angle in Phi direction.
esps	real	Stiffness/angle in Psi direction.
esr	real	Elastic stiffness for R stop and friction.
est	real	Stiffness/angle in Theta direction.
esx	real	Elastic stiffness for X stop and friction.
esy	real	Elastic stiffness for Y stop and friction.
esz	real	Elastic stiffness for Z stop and friction.
exists	logical	true if jstf exists, false if referred to but not defined. (read only)
ffr	integer	LC: Lim R force, or yield force vs R translation.
ffx	integer	LC: Lim X force, or yield force vs X translation.
ffy	integer	LC: Lim Y force, or yield force vs Y translation.
ffz	integer	LC: Lim Z force, or yield force vs Z translation.
fmal	integer	LC: Alpha Frictional moment vs rotation.

fmbt	integer	LC: Beta Frictional moment vs rotation.
fmph	integer	LC: Psi frictional moment vs rotation.
fmps	integer	LC: Psi frictional moment vs rotation.
fmt	integer	LC: Theta frictional moment vs rotation.
include	integer	The Include file number that the jstf is in.
jid	integer	Joint for restraint/table uses.
label	integer	JointStiffness ID of the JSTF. Also see the label property which is an alternative name for this.
lcidal	integer	LC: Alpha moment vs Rotation.
lcidbt	integer	LC: Beta moment vs Rotation.
lcidg	integer	LC: Gamma angle vs factor on Alpha blending.
lcidph	integer	LC: Phi moment vs rotation.
lcidps	integer	LC: Psi moment vs rotation.
lcidr	integer	LC: R force vs R rel displ.
lcidt	integer	LC: Theta moment vs rotation.
lcidx	integer	LC: X force vs X rel displ.
lcidy	integer	LC: Y force vs Y rel displ.
lcidz	integer	LC: Z force vs Z rel displ.
model	integer	The Model number that the joint stiffness is in.
nsabt	real	Stop angle for -ve Beta rotation.
nsaph	real	Stop angle for -ve Phi rotation.
nsaps	real	Stop angle for -ve Psi rotation.
nsat	real	Stop angle for -ve Theta rotation.
nsdx	real	Limiting -ve X translation.
nsdy	real	Limiting -ve Y translation.
nsdz	real	Limiting -ve Z translation.
option	constant	JointStiffness type. Can be JointStiffness.GENERALIZED, JointStiffness.FLEXION_TORSION, JointStiffness.TRANSLATIONAL or JointStiffness.CYLINDRICAL.
pida	integer	Part ID #A.
pidb	integer	Part ID #B.
psabt	real	Stop angle for +ve Beta rotation.
psaph	real	Stop angle for +ve Phi rotation.
psaps	real	Stop angle for +ve Psi rotation.
psat	real	Stop angle for +ve Theta rotation.
psdr	real	Limiting R translation.
psdx	real	Limiting +ve X translation.
psdy	real	Limiting +ve Y translation.
psdz	real	Limiting +ve Z translation.
rad1	real	Radius of pin.
rad2	real	Radius of hole.
saal	real	Stop angle for Alpha rotation.

Detailed Description

The JointStiffness class allows you to create, modify, edit and manipulate joint stiffness cards. See the documentation below for more details.

For convenience "Jstf" can also be used as the class name instead of "JointStiffness".

Constructor

new JointStiffness(Model*[Model]*, option*[constant]*, label*[integer]*, pida*[integer]*, pidb*[integer]*, cida*[integer]*, cidb*[integer]*, jid*[integer]*, lcidph*[integer]*, lcidt*[integer]*, lcidps*[integer]*, dlcidph*[integer]*, dlcidt*[integer]*, dlcidps*[integer]*, esph*[real]*, fmph*[integer]*, est*[real]*, fmt*[integer]*, esps*[real]*, fmps*[integer]*, nsaph*[real]*, psaph*[real]*, nsat*[real]*, psat*[real]*, nsaps*[real]*, psaps*[real]*)

Description

Create a new JointStiffness object for *CONSTRAINED_JOINT_STIFFNESS_GENERALIZED.

Arguments

Name	Type	Description
Model	Model	Model that jstf will be created in
option	constant	Must be JointStiffness.GENERALIZED.
label	integer	JointStiffness ID of the JSTF. Also see the label argument which is an alternative name for this.
pida	integer	Part ID #A.
pidb	integer	Part ID #B.
cida	integer	Coordinate System ID #A.
cidb	integer	Coordinate System ID #B.
jid	integer	Joint for restraint/table uses.
lcidph	integer	LC: Phi moment vs rotation.
lcidt	integer	LC: Theta moment vs rotation.
lcidps	integer	LC: Psi moment vs rotation.
dlcidph	integer	LC: Phi damping moment vs rotation vel.
dlcidt	integer	LC: Theta damping moment vs rotation vel.
dlcidps	integer	LC: Psi damping moment vs rotation vel.
esph	real	Stiffness/angle in Phi direction.
fmph	integer	LC: Psi frictional moment vs rotation.
est	real	Stiffness/angle in Theta direction.
fmt	integer	LC: Theta frictional moment vs rotation.
esps	real	Stiffness/angle in Psi direction.
fmps	integer	LC: Psi frictional moment vs rotation.
nsaph	real	Stop angle for -ve Phi rotation.
psaph	real	Stop angle for +ve Phi rotation.
nsat	real	Stop angle for -ve Theta rotation.

psat	real	Stop angle for +ve Theta rotation.
nsaps	real	Stop angle for -ve Psi rotation.
psaps	real	Stop angle for +ve Psi rotation.

Return type

JointStiffness object

Example

To create a new jstf 1000 of type GENERALIZED in model m with the following specifiction: pida, pidb, cida, cidb, jid are 91, 92, 81, 82, 71 respectively; lcidph, lcidt, lcidps, dlcidph, dlcidt, dlcidps are 1, 2, 3, 4, 5, 6 respectively; esph, fmph, est, fmt, esps, fmps are 11.0, 11, 12.0, 12, 13.0, 13 respectively; nsaph, psaph, nsat, psat, nsaps, psaps are -20, 20, -30, 30, -40, 40 respectively.

```
var j = new JointStiffness(m, JointStiffness.GENERALIZED, 1000, 91, 92, 81, 82,
71, 1, 2, 3, 4, 5, 6, 11.0, 11, 12.0, 12, 13.0, 13, -20, 20, -30, 30, -40, 40);
```

new JointStiffness(Model*[Model]*, option*[constant]*, label*[integer]*, pida*[integer]*, pidb*[integer]*, cida*[integer]*, cidb*[integer]*, jid*[integer]*, lcidal*[integer]*, lcidg*[integer]*, lcidbt*[integer]*, dlcidal*[integer]*, dlcidg*[integer]*, dlcidbt*[integer]*, esal*[real]*, fmal*[integer]*, esbt*[real]*, fmbt*[integer]*, saal*[real]*, nsabt*[real]*, psabt*[real]*)

Description

Create a new JointStiffness object for *CONSTRAINED_JOINT_STIFFNESS_FLEXION-TORSION.

Arguments

Name	Type	Description
Model	Model	Model that jstf will be created in
option	constant	Must be JointStiffness.FLEXION-TORSION.
label	integer	JointStiffness ID of the JSTF. Also see the label argument which is an alternative name for this.
pida	integer	Part ID #A.
pidb	integer	Part ID #B.
cida	integer	Coordinate System ID #A.
cidb	integer	Coordinate System ID #B.
jid	integer	Joint for restraint/table uses.
lcidal	integer	LC: Alpha moment vs Rotation.
lcidg	integer	LC: Gamma angle vs factor on Alpha blending.
lcidbt	integer	LC: Beta moment vs Rotation.
dlcidal	integer	LC: Alpha damping moment vs Rotl vel.
dlcidg	integer	LC: Gamma damping factor vs factor on Alpha damping moment.
dlcidbt	integer	LC: Beta damping moment vs Rotl vel.
esal	real	Stiffness/angle in Alpha direction.
fmal	integer	LC: Alpha Frictional moment vs rotation.
esbt	real	Stiffness/angle in Beta direction.
fmbt	integer	LC: Beta Frictional moment vs rotation.
saal	real	Stop angle for Alpha rotation.

nsabt	real	Stop angle for -ve Beta rotation.
psabt	real	Stop angle for +ve Beta rotation.

Return type

JointStiffness object

Example

To create a new jstf 2000 of type GENERALIZED in model m with the following specifiction: pida, pidb, cida, cidb, jid are 81, 82, 71, 72, 61 respectively; lcidal, lcidg, lcidbt, dlcidal, dlcidg, dlcidbt are 1, 2, 3, 4, 5, 6 respectively; esal, fmal, esbt, fmbt are 11.5, 12, 12.5, 13 respectively; saal, nsabt, psabt are 22.5, 25.0, 27.5 respectively.

```
var j = new JointStiffness(m, JointStiffness.FLEXION_TORSION, 2000, 81, 82, 71,
72, 61, 1, 2, 3, 4, 5, 6, 11.5, 12, 12.5, 13, 22.5, 25.0, 27.5);
```

new JointStiffness(Model*[Model]*, option*[constant]*, label*[integer]*, pida*[integer]*, pidb*[integer]*, cida*[integer]*, cidb*[integer]*, jid*[integer]*, lcidx*[integer]*, lcidy*[integer]*, lcidz*[integer]*, dlcidx*[integer]*, dlcidy*[integer]*, dlcidz*[integer]*, esx*[real]*, ffx*[integer]*, esy*[real]*, ffy*[integer]*, esz*[real]*, ffz*[integer]*, nsdx*[real]*, psdx*[real]*, nsdy*[real]*, psdy*[real]*, nsdz*[real]*, psdz*[real]*)

Description

Create a new JointStiffness object for *CONSTRAINED_JOINT_STIFFNESS_TRANSLATIONAL.

Arguments

Name	Type	Description
Model	Model	Model that jstf will be created in
option	constant	Must be JointStiffness.TRANSLATIONAL.
label	integer	JointStiffness ID of the JSTF. Also see the label argument which is an alternative name for this.
pida	integer	Part ID #A.
pidb	integer	Part ID #B.
cida	integer	Coordinate System ID #A.
cidb	integer	Coordinate System ID #B.
jid	integer	Joint for restraint/table uses.
lcidx	integer	LC: X force vs X rel displ.
lcidy	integer	LC: Y force vs Y rel displ.
lcidz	integer	LC: Z force vs Z rel displ.
dlcidx	integer	LC: X damping vs X rel velocity.
dlcidy	integer	LC: Y damping vs Y rel velocity.
dlcidz	integer	LC: Z damping vs Z rel velocity.
esx	real	Elastic stiffness for X stop and friction.
ffx	integer	LC: Lim X force, or yield force vs X translation.
esy	real	Elastic stiffness for Y stop and friction.
ffy	integer	LC: Lim Y force, or yield force vs Y translation.
esz	real	Elastic stiffness for Z stop and friction.
ffz	integer	LC: Lim Z force, or yield force vs Z translation.

nsdx	real	Limiting -ve X translation.
psdx	real	Limiting +ve X translation.
nsdy	real	Limiting -ve Y translation.
psdy	real	Limiting +ve Y translation.
nsdz	real	Limiting -ve Z translation.
psdz	real	Limiting +ve Z translation.

Return type

JointStiffness object

Example

To create a new jstf 3000 of type TRANSLATIONAL in model m with the following specifiction: pida, pidb, cida, cidb, jid are 71, 72, 61, 62, 51 respectively; lcidx, lcidy, lcidz, dlcidx, dlcidy, dlcidz are 1, 2, 3, 4, 5, 6 respectively; esx, ffx, esy, ffy, esz, ffz are are 12.5, 13, 13.5, 14, 14.5, 15 respectively; nsdx, psdx, nsdy, psdy, nsdz, psdz are -30, 30, -40, 40, -50, 50 respectively.

```
var j = new JointStiffness(m, JointStiffness.TRANSLATIONAL, 3000, 71, 72, 61,
62, 51, 1, 2, 3, 4, 5, 6, 12.5, 13, 13.5, 14, 14.5, 15, -30, 30, -40, 40, -50,
50);
```

new JointStiffness(Model*[Model]*, option*[constant]*, label*[integer]*, pida*[integer]*, pidb*[integer]*, cida*[integer]*, cidb*[integer]*, jid*[integer]*, lcidr*[integer]*, lcidz*[integer]*, dlcidr*[integer]*, dlcidp*[integer]*, dlcidz*[integer]*, lcidt*[integer]*, dlcidt*[integer]*, esr*[real]*, ffr*[integer]*, esz*[real]*, ffz*[integer]*, rad1*[real]*, rad2*[real]*, psdr*[real]*, nsdz*[real]*, psdz*[real]*)

Description

Create a new JointStiffness object for *CONSTRAINED_JOINT_STIFFNESS_CYLINDRICAL.

Arguments

Name	Type	Description
Model	Model	Model that jstf will be created in
option	constant	Must be JointStiffness.CYLINDRICAL.
label	integer	JointStiffness ID of the JSTF. Also see the label argument which is an alternative name for this.
pida	integer	Part ID #A.
pidb	integer	Part ID #B.
cida	integer	Coordinate System ID #A.
cidb	integer	Coordinate System ID #B.
jid	integer	Joint for restraint/table uses.
lcidr	integer	LC: R force vs R rel displ.
lcidz	integer	LC: Z force vs Z rel displ.
dlcidr	integer	LC: R damping vs R rel velocity.
dlcidp	integer	LC: P damping vs P rel velocity.
dlcidz	integer	LC: Z damping vs Z rel velocity.
lcidt	integer	LC: Theta moment vs rotation.
dlcidt	integer	LC: Theta damping moment vs rotation vel.

esr	real	Elastic stiffness for R stop and friction.
ffr	integer	LC: Lim R force, or yield force vs R translation.
esz	real	Elastic stiffness for Z stop and friction.
ffz	integer	LC: Lim Z force, or yield force vs Z translation.
rad1	real	Radius of pin.
rad2	real	Radius of hole.
psdr	real	Limiting R translation.
nsdz	real	Limiting -ve Z translation.
psdz	real	Limiting +ve Z translation.

Return type

JointStiffness object

Example

To create a new jstf 4000 of type CYLINDRICAL in model m with the following specifiction: pida, pidb, cida, cidb, jid are 61, 62, 51, 52, 41 respectively; lcidr, lcidz, dlcidr, dlcidp, dlcidz, lcidt, dlcidt are 1, 2, 3, 4, 5, 6, 7 respectively; esr, ffr, esz, ffz, rad1, rad2 are are 12.5, 13, 13.5, 14, 14.5, 15.5 respectively; psdr, nsdz, psdz are 30, -40, 50 respectively.

```
var j = new JointStiffness(m, JointStiffness.CYLINDRICAL, 4000, 61, 62, 51, 52, 41, 1, 2, 3, 4, 5, 6, 7, 12.5, 13, 13.5, 14, 14.5, 15.5, 30, -40, 50);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a joint stiffness.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the joint stiffness

Return type

No return value

Example

To associate comment c to the joint stiffness js:

```
js.AssociateComment(c);
```

Blank()

Description

Blanks the joint stiffness

Arguments

No arguments

Return type

No return value

Example

To blank joint stiffness js:

```
js.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the joint stiffnesses in model m:

```
JointStiffness.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joint stiffnesses will be blanked in
flag	Flag	Flag set on the joint stiffnesses that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the joint stiffnesses in model m flagged with f:

```
JointStiffness.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the joint stiffness is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if joint stiffness js is blanked:

```
if (js.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse joint stiffness js:

```
js.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the joint stiffness.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the joint stiffness

Return type

No return value

Example

To clear flag f for joint stiffness js:

```
js.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the joint stiffness.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

JointStiffness object

Example

To copy joint stiffness js into joint stiffness z:

```
var z = js.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a jstf.

Arguments

Name	Type	Description
Model	Model	Model that the jstf will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

JointStiffness object (or null if not made)

Example

To start creating a jstf in model m:

```
var m = JointStiffness.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a joint stiffness.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the joint stiffness

Return type

No return value

Example

To detach comment c from the joint stiffness js:

```
js.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit joint stiffness js:

```
js.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for joint stiffness. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for joint stiffness js:

```
js.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first joint stiffness in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first joint stiffness in

Return type

JointStiffness object (or null if there are no joint stiffnesses in the model).

Example

To get the first joint stiffness in model m:

```
var js = JointStiffness.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free joint stiffness label in the model. Also see JointStiffness.LastFreeLabel(), JointStiffness.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free joint stiffness label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

JointStiffness label.

Example

To get the first free joint stiffness label in model m:

```
var label = JointStiffness.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the joint stiffnesses in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be flagged in
flag	Flag	Flag to set on the joint stiffnesses

Return type

No return value

Example

To flag all of the joint stiffnesses with flag f in model m:

```
JointStiffness.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the joint stiffness is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the joint stiffness

Return type

true if flagged, false if not.

Example

To check if joint stiffness js has flag f set on it:

```
if (js.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each joint stiffness in the model.
Note that ForEach has been designed to make looping over joint stiffnesses as fast as possible and so has some limitations.
Firstly, a single temporary JointStiffness object is created and on each function call it is updated with the current joint stiffness data. This means that you should not try to store the JointStiffness object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new joint stiffnesses inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses are in
func	function	Function to call for each joint stiffness
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the joint stiffnesses in model m:

```
JointStiffness.ForEach(m, test);
function test(js)
{
// js is JointStiffness object
}
```

To call function test for all of the joint stiffnesses in model m with optional object:

```
var data = { x:0, y:0 };
JointStiffness.ForEach(m, test, data);
function test(js, extra)
{
// js is JointStiffness object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of JointStiffness objects for all of the joint stiffnesses in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get joint stiffnesses from

Return type

Array of JointStiffness objects

Example

To make an array of JointStiffness objects for all of the joint stiffnesses in model m

```
var js = JointStiffness.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a joint stiffness.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the joint stiffness js:

```
var comm_array = js.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of JointStiffness objects for all of the flagged joint stiffnesses in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get joint stiffnesses from
flag	Flag	Flag set on the joint stiffnesses that you want to retrieve

Return type

Array of JointStiffness objects

Example

To make an array of JointStiffness objects for all of the joint stiffnesses in model m flagged with f

```
var js = JointStiffness.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the JointStiffness object for a joint stiffness ID.

Arguments

Name	Type	Description
Model	Model	Model to find the joint stiffness in
number	integer	number of the joint stiffness you want the JointStiffness object for

Return type

JointStiffness object (or null if joint stiffness does not exist).

Example

To get the JointStiffness object for joint stiffness 100 in model m

```
var js = JointStiffness.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a JointStiffness property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the JointStiffness.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	joint stiffness property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if JointStiffness property js.example is a parameter:

```
Options.property_parameter_names = true;
if (js.GetParameter(js.example) ) do_something...
Options.property_parameter_names = false;
```

To check if JointStiffness property js.example is a parameter by using the GetParameter method:

```
if (js.ViewParameters().GetParameter(js.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this jstf (*CONSTRAINED_JOINT_STIFFNESS). **Note that a carriage return is not added**. See also JointStiffness.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for jstf n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the jstf. **Note that a carriage return is not added**. See also JointStiffness.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for jstf n:

```
var cards = n.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last joint stiffness in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last joint stiffness in

Return type

JointStiffness object (or null if there are no joint stiffnesses in the model).

Example

To get the last joint stiffness in model m:

```
var js = JointStiffness.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free joint stiffness label in the model. Also see JointStiffness.FirstFreeLabel(), JointStiffness.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free joint stiffness label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

JointStiffness label.

Example

To get the last free joint stiffness label in model m:

```
var label = JointStiffness.LastFreeLabel(m);
```

Next()

Description

Returns the next joint stiffness in the model.

Arguments

No arguments

Return type

JointStiffness object (or null if there are no more joint stiffnesses in the model).

Example

To get the joint stiffness in model m after joint stiffness js:

```
var js = js.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) joint stiffness label in the model. Also see JointStiffness.FirstFreeLabel(), JointStiffness.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free joint stiffness label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

JointStiffness label.

Example

To get the next free joint stiffness label in model m:

```
var label = JointStiffness.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a joint stiffness.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only joint stiffnesses from that model can be picked. If the argument is a Flag then only joint stiffnesses that are flagged with *limit* can be selected. If omitted, or null, any joint stiffnesses from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

JointStiffness object (or null if not picked)

Example

To pick a joint stiffness from model m giving the prompt 'Pick joint stiffness from screen':

```
var js = JointStiffness.Pick('Pick joint stiffness from screen', m);
```

Previous()

Description

Returns the previous joint stiffness in the model.

Arguments

No arguments

Return type

JointStiffness object (or null if there are no more joint stiffnesses in the model).

Example

To get the joint stiffness in model m before joint stiffness js:

```
var js = js.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the joint stiffnesses in model m, from 1000000:

```
JointStiffness.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joint stiffnesses will be renumbered in
flag	Flag	Flag set on the joint stiffnesses that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the joint stiffnesses in model m flagged with f, from 1000000:

```
JointStiffness.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select joint stiffnesses using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting joint stiffnesses
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only joint stiffnesses from that model can be selected. If the argument is a Flag then only joint stiffnesses that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any joint stiffnesses can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of joint stiffnesses selected or null if menu cancelled

Example

To select joint stiffnesses from model m, flagging those selected with flag f, giving the prompt 'Select joint stiffnesses':

```
JointStiffness.Select(f, 'Select joint stiffnesses', m);
```

To select joint stiffnesses, flagging those selected with flag f but limiting selection to joint stiffnesses flagged with flag l, giving the prompt 'Select joint stiffnesses':

```
JointStiffness.Select(f, 'Select joint stiffnesses', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the joint stiffness.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the joint stiffness

Return type

No return value

Example

To set flag f for joint stiffness js:

```
js.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the joint stiffness. The joint stiffness will be sketched until you either call JointStiffness.Unsketch(), JointStiffness.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the joint stiffness is sketched. If omitted redraw is true. If you want to sketch several joint stiffnesses and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch joint stiffness js:

```
js.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged joint stiffnesses in the model. The joint stiffnesses will be sketched until you either call JointStiffness.Unsketch(), JointStiffness.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged joint stiffnesses will be sketched in
flag	Flag	Flag set on the joint stiffnesses that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the joint stiffnesses are sketched. If omitted redraw is true. If you want to sketch flagged joint stiffnesses several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all joint stiffnesses flagged with flag in model m:

```
JointStiffness.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing joint stiffnesses should be counted. If false or omitted referenced but undefined joint stiffnesses will also be included in the total.

Return type

number of joint stiffnesses

Example

To get the total number of joint stiffnesses in model m:

```
var total = JointStiffness.Total(m);
```

Unblank()

Description

Unblanks the joint stiffness

Arguments

No arguments

Return type

No return value

Example

To unblank joint stiffness js:

```
js.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the joint stiffnesses in model m:

```
JointStiffness.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged joint stiffnesses will be unblanked in
flag	Flag	Flag set on the joint stiffnesses that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the joint stiffnesses in model m flagged with f:

```
JointStiffness.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all joint stiffnesses will be unset in

flag	Flag	Flag to unset on the joint stiffnesses

Return type

No return value

Example

To unset the flag f on all the joint stiffnesses in model m:

```
JointStiffness.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the joint stiffness.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the joint stiffness is unsketched. If omitted redraw is true. If you want to unsketch several joint stiffnesses and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch joint stiffness js:

```
js.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all joint stiffnesses.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the joint stiffnesses are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all joint stiffnesses in model m:

```
JointStiffness.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged joint stiffnesses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all joint stiffnesses will be unsketched in
flag	Flag	Flag set on the joint stiffnesses that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the joint stiffnesses are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all joint stiffnesses flagged with flag in model m:

```
JointStiffness.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

JointStiffness object.

Example

To check if JointStiffness property js.example is a parameter by using the JointStiffness.GetParameter() method:

```
if (js.ViewParameters().GetParameter(js.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for joint stiffness. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for joint stiffness js:

```
js.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this joint stiffness.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for joint stiffness js:

```
var xrefs = js.Xrefs();
```

---

toString()

Description

Creates a string containing the jstf data in keyword format. Note that this contains the keyword header and the keyword cards. See also JointStiffness.Keyword() and JointStiffness.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for jstf n in keyword format

```
var s = n.toString();
```

Linear class

The Linear class gives you access to define *CONSTRAINED_LINEAR cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AddRowData(nid*[integer]*, dof*[integer]*, coeff*[real]*, cid (Optional)*[integer]*)
- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRowData(row_index*[Integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRowData(row_index*[Integer]*)
- SetFlag(flag*[Flag]*)
- SetRowData(row_index*[Integer]*, nid*[integer]*, dof*[integer]*, coeff*[real]*, cid (Optional)*[integer]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)

- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Linear constants

Name	Description
Linear.GLOBAL	CNST is *CONSTRAINED_LINEAR_GLOBAL.
Linear.LOCAL	CNST is *CONSTRAINED_LINEAR_LOCAL.

Linear properties

Name	Type	Description
exists (read only)	logical	true if Linear exists, false if referred to but not defined. (read only)
format	constant	The Constrained Linear option. Can be Linear.GLOBAL or Linear.LOCAL.
include	integer	The Include file number that the Linear is in.
lcid	integer	Linear label
model	integer	The Model number that the constrained linear is in.

Detailed Description

The Linear class allows you to create, modify, edit and manipulate *CONSTRAINED_LINEAR cards. See the documentation below for more details.

Constructor

new Linear(Model*[Model]*, format*[constant]*, lcid*[integer]*, nid*[integer]*, dof*[integer]*, coeff*[real]*, cid (Optional)*[integer]*)

Description

Create a new Linear object.

Arguments

Name	Type	Description
Model	Model	Model that Linear will be created in
format	constant	Specify the type of constrained linear. Can be Linear.GLOBAL or Linear.LOCAL)
lcid	integer	Linear label
nid	integer	Node id.
dof	integer	Degrees-of-Freedom.
coeff	real	Non-zero coefficient.
cid (Optional)	integer	Coordinate System ID if format is Linear.LOCAL. The default value is 0.

Return type

Linear object

Example

To create a new constrained linear in model m of type LOCAL with lcid 1, nid 1, dof 3, coeff 0.5 and cid 2

```
var c_l = new Linear(m,Linear.LOCAL,1,1,3,0.5,2);
```

Details of functions

AddRowData(nid*[integer]*, dof*[integer]*, coeff*[real]*, cid (Optional)*[integer]*)

Description

Used to add additional independent card 2 to the keyword. Adds this data to the end of the selected *CONSTRAINED_LINEAR

Arguments

Name	Type	Description
nid	integer	Node id.
dof	integer	Degrees-of-Freedom.
coeff	real	Non-zero coefficient.
cid (Optional)	integer	Coordinate System ID if format is Linear.LOCAL. The default value is 0.

Return type

No return value

Example

To add NID 10 to the keyword c_l with dof 4, coeff 1.3, cid 2:

```
c_l.AddRowData(10,4,1.3,2);
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a constrained linear.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the constrained linear

Return type

No return value

Example

To associate comment c to the constrained linear c_l:

```
c_l.AssociateComment(c);
```

Blank()

Description

Blanks the constrained linear

Arguments

No arguments

Return type

No return value

Example

To blank constrained linear c_l:

```
c_l.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained linears in model m:

```
Linear.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained linears will be blanked in
flag	Flag	Flag set on the constrained linears that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the constrained linears in model m flagged with f:

```
Linear.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the constrained linear is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if constrained linear c_l is blanked:

```
if (c_l.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse constrained linear c_l:

```
c_l.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the constrained linear.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the constrained linear

Return type

No return value

Example

To clear flag f for constrained linear c_l:

```
c_l.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the constrained linear.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Linear object

Example

To copy constrained linear c_l into constrained linear z:

```
var z = c_l.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a Linear.

Arguments

Name	Type	Description
Model	Model	Model that the constrainedLinear will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Linear object (or null if not made)

Example

To start creating a constrainedLinear in model m:

```
var c_l = Linear.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a constrained linear.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the constrained linear

Return type

No return value

Example

To detach comment c from the constrained linear c_l:

```
c_l.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit constrained linear c_l:

```
c_l.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for constrained linear. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for constrained linear c_l:

```
c_l.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first constrained linear in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first constrained linear in

Return type

Linear object (or null if there are no constrained linears in the model).

Example

To get the first constrained linear in model m:

```
var c_l = Linear.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free constrained linear label in the model. Also see Linear.LastFreeLabel(), Linear.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free constrained linear label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Linear label.

Example

To get the first free constrained linear label in model m:

```
var label = Linear.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the constrained linears in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be flagged in
flag	Flag	Flag to set on the constrained linears

Return type

No return value

Example

To flag all of the constrained linears with flag f in model m:

```
Linear.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the constrained linear is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the constrained linear

Return type

true if flagged, false if not.

Example

To check if constrained linear c_l has flag f set on it:

```
if (c_l.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each constrained linear in the model.
Note that ForEach has been designed to make looping over constrained linears as fast as possible and so has some limitations.
Firstly, a single temporary Linear object is created and on each function call it is updated with the current constrained linear data. This means that you should not try to store the Linear object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new constrained linears inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears are in
func	function	Function to call for each constrained linear
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the constrained linears in model m:

```
Linear.ForEach(m, test);
function test(c_l)
{
// c_l is Linear object
}
```

To call function test for all of the constrained linears in model m with optional object:

```
var data = { x:0, y:0 };
Linear.ForEach(m, test, data);
function test(c_l, extra)
{
// c_l is Linear object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Linear objects for all of the constrained linears in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained linears from

Return type

Array of Linear objects

Example

To make an array of Linear objects for all of the constrained linears in model m

```
var c_l = Linear.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a constrained linear.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the constrained linear c_l:

```
var comm_array = c_l.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of Linear objects for all of the flagged constrained linears in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get constrained linears from
flag	Flag	Flag set on the constrained linears that you want to retrieve

Return type

Array of Linear objects

Example

To make an array of Linear objects for all of the constrained linears in model m flagged with f

```
var c_l = Linear.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Linear object for a constrained linear ID.

Arguments

Name	Type	Description
Model	Model	Model to find the constrained linear in
number	integer	number of the constrained linear you want the Linear object for

Return type

Linear object (or null if constrained linear does not exist).

Example

To get the Linear object for constrained linear 100 in model m

```
var c_l = Linear.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Linear property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Linear.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	constrained linear property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Linear property c_l.example is a parameter:

```
Options.property_parameter_names = true;
if (c_l.GetParameter(c_l.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Linear property c_l.example is a parameter by using the GetParameter method:

```
if (c_l.ViewParameters().GetParameter(c_l.example) ) do_something...
```

GetRowData(row_index*[Integer]*)

Description

Returns independent card 2 for the selected row of the *CONSTRAINED_LINEAR.

Arguments

Name	Type	Description
row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Linear.total

Return type

Array containing data.

Example

To loop over all the lines of the keyword for c_l:

```
for (i=0; i<c_l.total; i++)
        var data = c_l.GetRowData(i);
```

Keyword()

Description

Returns the keyword for this Linear (*constrained_linear). **Note that a carriage return is not added**. See also Linear.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Linear c_l:

```
var key = c_l.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Linear. **Note that a carriage return is not added**. See also Linear.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Linear c_l:

```
var cards = c_l.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last constrained linear in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last constrained linear in

Return type

Linear object (or null if there are no constrained linears in the model).

Example

To get the last constrained linear in model m:

```
var c_l = Linear.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free constrained linear label in the model. Also see Linear.FirstFreeLabel(), Linear.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free constrained linear label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Linear label.

Example

To get the last free constrained linear label in model m:

```
var label = Linear.LastFreeLabel(m);
```

Next()

Description

Returns the next constrained linear in the model.

Arguments

No arguments

Return type

Linear object (or null if there are no more constrained linears in the model).

Example

To get the constrained linear in model m after constrained linear c_l:

```
var c_l = c_l.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) constrained linear label in the model. Also see Linear.FirstFreeLabel(), Linear.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free constrained linear label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Linear label.

Example

To get the next free constrained linear label in model m:

```
var label = Linear.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a constrained linear.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained linears from that model can be picked. If the argument is a Flag then only constrained linears that are flagged with *limit* can be selected. If omitted, or null, any constrained linears from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Linear object (or null if not picked)

Example

To pick a constrained linear from model m giving the prompt 'Pick constrained linear from screen':

```
var c_l = Linear.Pick('Pick constrained linear from screen', m);
```

Previous()

Description

Returns the previous constrained linear in the model.

Arguments

No arguments

Return type

Linear object (or null if there are no more constrained linears in the model).

Example

To get the constrained linear in model m before constrained linear c_l:

```
var c_l = c_l.Previous();
```

RemoveRowData(row_index*[Integer]*)

Description

Removes an independent card 2 for the selected row on the *CONSTRAINED_LINEAR.

Arguments

Name	Type	Description
row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Linear.total

Return type

No return value.

Example

To remove row 2 for c_l:

```
c_l.RemoveRowData(1);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the constrained linears in model m, from 1000000:

```
Linear.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained linears will be renumbered in
flag	Flag	Flag set on the constrained linears that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the constrained linears in model m flagged with f, from 1000000:

```
Linear.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select constrained linears using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting constrained linears
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only constrained linears from that model can be selected. If the argument is a Flag then only constrained linears that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any constrained linears can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of constrained linears selected or null if menu cancelled

Example

To select constrained linears from model m, flagging those selected with flag f, giving the prompt 'Select constrained linears':

```
Linear.Select(f, 'Select constrained linears', m);
```

To select constrained linears, flagging those selected with flag f but limiting selection to constrained linears flagged with flag l, giving the prompt 'Select constrained linears':

```
Linear.Select(f, 'Select constrained linears', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the constrained linear.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the constrained linear

Return type

No return value

Example

To set flag f for constrained linear c_l:

```
c_l.SetFlag(f);
```

SetRowData(row_index*[Integer]*, nid*[integer]*, dof*[integer]*, coeff*[real]*, cid (Optional)*[integer]*)

Description

Used to reset values in already existing card 2 in the selected row of *CONSTRAINED_LINEAR

Arguments

Name	Type	Description
row_index	Integer	The row index of the data to return. **Note that indices start at 0, not 1**. 0 <= row_index < Linear.total
nid	integer	Node id.
dof	integer	Degrees-of-Freedom.
coeff	real	Non-zero coefficient.
cid (Optional)	integer	Coordinate System ID if format is Linear.LOCAL. The default value is 0.

Return type

No return value

Example

To reset the values of row 3 of the keyword with NID 11, dof 2, coeff 3.2, cid 4:

```
c_l.SetRowData(2,11,2,3.2,4);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the constrained linear. The constrained linear will be sketched until you either call Linear.Unsketch(), Linear.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained linear is sketched. If omitted redraw is true. If you want to sketch several constrained linears and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch constrained linear c_l:

```
c_l.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged constrained linears in the model. The constrained linears will be sketched until you either call Linear.Unsketch(), Linear.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged constrained linears will be sketched in
flag	Flag	Flag set on the constrained linears that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the constrained linears are sketched. If omitted redraw is true. If you want to sketch flagged constrained linears several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all constrained linears flagged with flag in model m:

```
Linear.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing constrained linears should be counted. If false or omitted referenced but undefined constrained linears will also be included in the total.

Return type

number of constrained linears

Example

To get the total number of constrained linears in model m:

```
var total = Linear.Total(m);
```

Unblank()

Description

Unblanks the constrained linear

Arguments

No arguments

Return type

No return value

Example

To unblank constrained linear c_l:

```
c_l.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained linears in model m:

```
Linear.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged constrained linears will be unblanked in
flag	Flag	Flag set on the constrained linears that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the constrained linears in model m flagged with f:

```
Linear.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all constrained linears will be unset in
flag	Flag	Flag to unset on the constrained linears

Return type

No return value

Example

To unset the flag f on all the constrained linears in model m:

```
Linear.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the constrained linear.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the constrained linear is unsketched. If omitted redraw is true. If you want to unsketch several constrained linears and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch constrained linear c_l:

```
c_l.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all constrained linears.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the constrained linears are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained linears in model m:

```
Linear.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged constrained linears in the model.

Arguments

Name	Type	Description
Model	Model	Model that all constrained linears will be unsketched in
flag	Flag	Flag set on the constrained linears that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the constrained linears are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all constrained linears flagged with flag in model m:

```
Linear.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Linear object.

Example

To check if Linear property c_l.example is a parameter by using the Linear.GetParameter() method:

```
if (c_l.ViewParameters().GetParameter(c_l.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for constrained linear. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for constrained linear c_l:

```
c_l.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this constrained linear.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for constrained linear c_l:

```
var xrefs = c_l.Xrefs();
```

---

toString()

Description

Creates a string containing the Linear data in keyword format. Note that this contains the keyword header and the keyword cards. See also Linear.Keyword() and Linear.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Linear c_l in keyword format

```
var s = c_l.toString();
```

NodalRigidBody (Nrb) class

The NodalRigidBody class gives you access to define nodal rigid body cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

NodalRigidBody properties

Name	Type	Description
cid	integer	Coordinate system ID
cmo	integer	Centre of mass option
con1	integer	First retraint parameter
con2	integer	Second retraint parameter
drflag	integer	Displacement release flag
exists	logical	true if nrb exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the nrb is in.
inertia	logical	Flag to turn on or off _INERTIA option
iprt	integer	Print flag
ixx	real	Ixx component of inertia tensor
ixy	real	Ixy component of inertia tensor
ixz	real	Ixz component of inertia tensor
iyy	real	Iyy component of inertia tensor
iyz	real	Iyz component of inertia tensor
izz	real	Izz component of inertia tensor
label	integer	NodalRigidBody ID of the NRB. Also see the pid property which is an alternative name for this.
model	integer	The Model number that the nodal rigid body is in.
nodeid	integer	Optional node point
nsid	integer	Nodal set ID
override	logical	Flag to turn on or off _OVERRIDE option
pid	integer	NodalRigidBody ID of the NRB. Also see the label property which is an alternative name for this.
pnode	integer	Optional nodal point
rrflag	integer	Rotation release flag
spc	logical	Flag to turn on or off _SPC option
tm	real	Translational mass
vrx	real	X rigid body rotational velocity
vry	real	Y rigid body rotational velocity
vrz	real	Z rigid body rotational velocity
vtx	real	X rigid body translational velocity
vty	real	Y rigid body translational velocity
vtz	real	Z rigid body translational velocity
xc	real	X coordinate centre of mass
xl	real	X coordinate of local x axis
xlip	real	X coordinate of local in plane vector
yc	real	Y coordinate centre of mass
yl	real	Y coordinate of local x axis

ylip	real	Y coordinate of local in plane vector
zc	real	Z coordinate centre of mass
zl	real	Z coordinate of local x axis
zlip	real	Z coordinate of local in plane vector

Detailed Description

The NodalRigidBody class allows you to create, modify, edit and manipulate nodal rigid body cards. See the documentation below for more details.

For convenience "Nrb" can also be used as the class name instead of "NodalRigidBody".

Constructor

new NodalRigidBody(Model*[Model]*, nsid*[integer]*, pid (optional)*[integer]*, cid (optional)*[integer]*, pnode (optional)*[integer]*, iprt (optional)*[integer]*, drflag (optional)*[integer]*, rrflag (optional)*[integer]*)

Description

Create a new NodalRigidBody object.

Arguments

Name	Type	Description
Model	Model	Model that nrb will be created in
nsid	integer	Nodal set ID
pid (optional)	integer	NodalRigidBody ID of the NRB. Also see the label property which is an alternative name for this.
cid (optional)	integer	Coordinate system ID
pnode (optional)	integer	Optional nodal point
iprt (optional)	integer	Print flag
drflag (optional)	integer	Displacement release flag
rrflag (optional)	integer	Rotation release flag

Return type

NodalRigidBody object

Example

To create a new nrb in model m with label 200, using node set 50

```
var v = new NodalRigidBody(m, 50, 200);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a nodal rigid body.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the nodal rigid body

Return type

No return value

Example

To associate comment c to the nodal rigid body nrb:

```
nrb.AssociateComment(c);
```

Blank()

Description

Blanks the nodal rigid body

Arguments

No arguments

Return type

No return value

Example

To blank nodal rigid body nrb:

```
nrb.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodal rigid bodies in model m:

```
NodalRigidBody.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodal rigid bodies will be blanked in
flag	Flag	Flag set on the nodal rigid bodies that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodal rigid bodies in model m flagged with f:

```
NodalRigidBody.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the nodal rigid body is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if nodal rigid body nrb is blanked:

```
if (nrb.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse nodal rigid body nrb:

```
nrb.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the nodal rigid body.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the nodal rigid body

Return type

No return value

Example

To clear flag f for nodal rigid body nrb:

```
nrb.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the nodal rigid body.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

NodalRigidBody object

Example

To copy nodal rigid body nrb into nodal rigid body z:

```
var z = nrb.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a nrb.

Arguments

Name	Type	Description
Model	Model	Model that the nrb will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

NodalRigidBody object (or null if not made)

Example

To start creating a nrb in model m:

```
var m = NodalRigidBody.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a nodal rigid body.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the nodal rigid body

Return type

No return value

Example

To detach comment c from the nodal rigid body nrb:

```
nrb.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit nodal rigid body nrb:

```
nrb.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for nodal rigid body. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for nodal rigid body nrb:

```
nrb.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first nodal rigid body in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first nodal rigid body in

Return type

NodalRigidBody object (or null if there are no nodal rigid bodies in the model).

Example

To get the first nodal rigid body in model m:

```
var nrb = NodalRigidBody.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free nodal rigid body label in the model. Also see NodalRigidBody.LastFreeLabel(), NodalRigidBody.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free nodal rigid body label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

NodalRigidBody label.

Example

To get the first free nodal rigid body label in model m:

```
var label = NodalRigidBody.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the nodal rigid bodies in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be flagged in
flag	Flag	Flag to set on the nodal rigid bodies

Return type

No return value

Example

To flag all of the nodal rigid bodies with flag f in model m:

```
NodalRigidBody.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the nodal rigid body is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the nodal rigid body

Return type

true if flagged, false if not.

Example

To check if nodal rigid body nrb has flag f set on it:

```
if (nrb.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each nodal rigid body in the model.
Note that ForEach has been designed to make looping over nodal rigid bodies as fast as possible and so has some limitations.
Firstly, a single temporary NodalRigidBody object is created and on each function call it is updated with the current nodal rigid body data. This means that you should not try to store the NodalRigidBody object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new nodal rigid bodies inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies are in
func	function	Function to call for each nodal rigid body
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the nodal rigid bodies in model m:

```
NodalRigidBody.ForEach(m, test);
function test(nrb)
{
// nrb is NodalRigidBody object
}
```

To call function test for all of the nodal rigid bodies in model m with optional object:

```
var data = { x:0, y:0 };
NodalRigidBody.ForEach(m, test, data);
function test(nrb, extra)
{
// nrb is NodalRigidBody object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of NodalRigidBody objects for all of the nodal rigid bodies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodal rigid bodies from

Return type

Array of NodalRigidBody objects

Example

To make an array of NodalRigidBody objects for all of the nodal rigid bodies in model m

```
var nrb = NodalRigidBody.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a nodal rigid body.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the nodal rigid body nrb:

```
var comm_array = nrb.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of NodalRigidBody objects for all of the flagged nodal rigid bodies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodal rigid bodies from
flag	Flag	Flag set on the nodal rigid bodies that you want to retrieve

Return type

Array of NodalRigidBody objects

Example

To make an array of NodalRigidBody objects for all of the nodal rigid bodies in model m flagged with f

```
var nrb = NodalRigidBody.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the NodalRigidBody object for a nodal rigid body ID.

Arguments

Name	Type	Description
Model	Model	Model to find the nodal rigid body in
number	integer	number of the nodal rigid body you want the NodalRigidBody object for

Return type

NodalRigidBody object (or null if nodal rigid body does not exist).

Example

To get the NodalRigidBody object for nodal rigid body 100 in model m

```
var nrb = NodalRigidBody.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a NodalRigidBody property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the NodalRigidBody.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	nodal rigid body property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if NodalRigidBody property nrb.example is a parameter:

```
Options.property_parameter_names = true;
if (nrb.GetParameter(nrb.example) ) do_something...
Options.property_parameter_names = false;
```

To check if NodalRigidBody property nrb.example is a parameter by using the GetParameter method:

```
if (nrb.ViewParameters().GetParameter(nrb.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this nrb (*CONSTRAINED_NODAL_RIGID_BODY_xxxx). **Note that a carriage return is not added**. See also NodalRigidBody.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for nrb n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the nrb. **Note that a carriage return is not added**. See also NodalRigidBody.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for nrb n:

```
var cards = n.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last nodal rigid body in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last nodal rigid body in

Return type

NodalRigidBody object (or null if there are no nodal rigid bodies in the model).

Example

To get the last nodal rigid body in model m:

```
var nrb = NodalRigidBody.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free nodal rigid body label in the model. Also see NodalRigidBody.FirstFreeLabel(), NodalRigidBody.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free nodal rigid body label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

NodalRigidBody label.

Example

To get the last free nodal rigid body label in model m:

```
var label = NodalRigidBody.LastFreeLabel(m);
```

Next()

Description

Returns the next nodal rigid body in the model.

Arguments

No arguments

Return type

NodalRigidBody object (or null if there are no more nodal rigid bodies in the model).

Example

To get the nodal rigid body in model m after nodal rigid body nrb:

```
var nrb = nrb.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) nodal rigid body label in the model. Also see NodalRigidBody.FirstFreeLabel(), NodalRigidBody.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free nodal rigid body label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

NodalRigidBody label.

Example

To get the next free nodal rigid body label in model m:

```
var label = NodalRigidBody.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a nodal rigid body.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodal rigid bodies from that model can be picked. If the argument is a Flag then only nodal rigid bodies that are flagged with *limit* can be selected. If omitted, or null, any nodal rigid bodies from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

NodalRigidBody object (or null if not picked)

Example

To pick a nodal rigid body from model m giving the prompt 'Pick nodal rigid body from screen':

```
var nrb = NodalRigidBody.Pick('Pick nodal rigid body from screen', m);
```

Previous()

Description

Returns the previous nodal rigid body in the model.

Arguments

No arguments

Return type

NodalRigidBody object (or null if there are no more nodal rigid bodies in the model).

Example

To get the nodal rigid body in model m before nodal rigid body nrb:

```
var nrb = nrb.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the nodal rigid bodies in model m, from 1000000:

```
NodalRigidBody.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodal rigid bodies will be renumbered in
flag	Flag	Flag set on the nodal rigid bodies that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the nodal rigid bodies in model m flagged with f, from 1000000:

```
NodalRigidBody.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select nodal rigid bodies using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting nodal rigid bodies
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodal rigid bodies from that model can be selected. If the argument is a Flag then only nodal rigid bodies that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any nodal rigid bodies can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of nodal rigid bodies selected or null if menu cancelled

Example

To select nodal rigid bodies from model m, flagging those selected with flag f, giving the prompt 'Select nodal rigid bodies':

```
NodalRigidBody.Select(f, 'Select nodal rigid bodies', m);
```

To select nodal rigid bodies, flagging those selected with flag f but limiting selection to nodal rigid bodies flagged with flag l, giving the prompt 'Select nodal rigid bodies':

```
NodalRigidBody.Select(f, 'Select nodal rigid bodies', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the nodal rigid body.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the nodal rigid body

Return type

No return value

Example

To set flag f for nodal rigid body nrb:

```
nrb.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the nodal rigid body. The nodal rigid body will be sketched until you either call NodalRigidBody.Unsketch(), NodalRigidBody.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the nodal rigid body is sketched. If omitted redraw is true. If you want to sketch several nodal rigid bodies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch nodal rigid body nrb:

```
nrb.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged nodal rigid bodies in the model. The nodal rigid bodies will be sketched until you either call NodalRigidBody.Unsketch(), NodalRigidBody.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodal rigid bodies will be sketched in
flag	Flag	Flag set on the nodal rigid bodies that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the nodal rigid bodies are sketched. If omitted redraw is true. If you want to sketch flagged nodal rigid bodies several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all nodal rigid bodies flagged with flag in model m:

```
NodalRigidBody.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing nodal rigid bodies should be counted. If false or omitted referenced but undefined nodal rigid bodies will also be included in the total.

Return type

number of nodal rigid bodies

Example

To get the total number of nodal rigid bodies in model m:

```
var total = NodalRigidBody.Total(m);
```

Unblank()

Description

Unblanks the nodal rigid body

Arguments

No arguments

Return type

No return value

Example

To unblank nodal rigid body nrb:

```
nrb.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodal rigid bodies in model m:

```
NodalRigidBody.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged nodal rigid bodies will be unblanked in
flag	Flag	Flag set on the nodal rigid bodies that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodal rigid bodies in model m flagged with f:

```
NodalRigidBody.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all nodal rigid bodies will be unset in
flag	Flag	Flag to unset on the nodal rigid bodies

Return type

No return value

Example

To unset the flag f on all the nodal rigid bodies in model m:

```
NodalRigidBody.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the nodal rigid body.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the nodal rigid body is unsketched. If omitted redraw is true. If you want to unsketch several nodal rigid bodies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch nodal rigid body nrb:

```
nrb.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all nodal rigid bodies.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the nodal rigid bodies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodal rigid bodies in model m:

```
NodalRigidBody.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged nodal rigid bodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal rigid bodies will be unsketched in
flag	Flag	Flag set on the nodal rigid bodies that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the nodal rigid bodies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodal rigid bodies flagged with flag in model m:

```
NodalRigidBody.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

NodalRigidBody object.

Example

To check if NodalRigidBody property nrb.example is a parameter by using the NodalRigidBody.GetParameter() method:

```
if (nrb.ViewParameters().GetParameter(nrb.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for nodal rigid body. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for nodal rigid body nrb:

```
nrb.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this nodal rigid body.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for nodal rigid body nrb:

```
var xrefs = nrb.Xrefs();
```

---

toString()

Description

Creates a string containing the nrb data in keyword format. Note that this contains the keyword header and the keyword cards. See also NodalRigidBody.Keyword() and NodalRigidBody.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for nrb n in keyword format

```
var s = n.toString();
```

NodeSet class

The NodeSet class gives you access to constrained node set cards in PRIMER, **not** set node cards. For access to set node cards, refer to the Set class. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

NodeSet properties

Name	Type	Description
cnsid	integer	Constrained node set number (identical to label).
dof	integer	Degree of freedom.
exists	logical	true if constrained node set exists, false if referred to but not defined. (read only)
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the constrained node set is in.
label	integer	Constrained node set number.
model	integer	The Model number that the node set is in.
nsid	integer	Set Node ID.
tf	real	Failure time.

Detailed Description

The NodeSet class allows you to create, modify, edit and manipulate constrained node set cards. See the documentation below for more details.

Constructor

new NodeSet(Model*[Model]*, nsid*[integer]*, dof*[integer]*, tf*[real]*, label (optional)*[integer]*)

Description

Create a new NodeSet object.

Arguments

Name	Type	Description
Model	Model	Model that constrained node set will be created in
nsid	integer	Set Node ID.
dof	integer	Degree of freedom.
tf	real	Failure time.
label (optional)	integer	Constrained node set number.

Return type

NodeSet object

Example

To create a new constrained node set 500 in model m, of type SET, with node set 9, degree of freedom 1 and failure time 1000

```
var n = new NodeSet(m, 9, 1, 1000, 500);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a node set.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the node set

Return type

No return value

Example

To associate comment c to the node set ns:

```
ns.AssociateComment(c);
```

Blank()

Description

Blanks the node set

Arguments

No arguments

Return type

No return value

Example

To blank node set ns:

```
ns.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the node sets in model m:

```
NodeSet.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged node sets will be blanked in
flag	Flag	Flag set on the node sets that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the node sets in model m flagged with f:

```
NodeSet.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the node set is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if node set ns is blanked:

```
if (ns.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse node set ns:

```
ns.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the node set.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the node set

Return type

No return value

Example

To clear flag f for node set ns:

```
ns.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the node set.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

NodeSet object

Example

To copy node set ns into node set z:

```
var z = ns.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a node_set.

Arguments

Name	Type	Description
Model	Model	Model that the node_set will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

NodeSet object (or null if not made)

Example

To start creating a node set in model n:

```
var n = NodeSet.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a node set.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the node set

Return type

No return value

Example

To detach comment c from the node set ns:

```
ns.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit node set ns:

```
ns.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for node set. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for node set ns:

```
ns.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first node set in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first node set in

Return type

NodeSet object (or null if there are no node sets in the model).

Example

To get the first node set in model m:

```
var ns = NodeSet.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free node set label in the model. Also see NodeSet.LastFreeLabel(), NodeSet.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free node set label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

NodeSet label.

Example

To get the first free node set label in model m:

```
var label = NodeSet.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the node sets in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be flagged in
flag	Flag	Flag to set on the node sets

Return type

No return value

Example

To flag all of the node sets with flag f in model m:

```
NodeSet.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the node set is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the node set

Return type

true if flagged, false if not.

Example

To check if node set ns has flag f set on it:

```
if (ns.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each node set in the model.
Note that ForEach has been designed to make looping over node sets as fast as possible and so has some limitations.
Firstly, a single temporary NodeSet object is created and on each function call it is updated with the current node set data. This means that you should not try to store the NodeSet object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new node sets inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all node sets are in
func	function	Function to call for each node set
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the node sets in model m:

```
NodeSet.ForEach(m, test);
function test(ns)
{
// ns is NodeSet object
}
```

To call function test for all of the node sets in model m with optional object:

```
var data = { x:0, y:0 };
NodeSet.ForEach(m, test, data);
function test(ns, extra)
{
// ns is NodeSet object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of NodeSet objects for all of the node sets in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get node sets from

Return type

Array of NodeSet objects

Example

To make an array of NodeSet objects for all of the node sets in model m

```
var ns = NodeSet.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a node set.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the node set ns:

```
var comm_array = ns.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of NodeSet objects for all of the flagged node sets in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get node sets from
flag	Flag	Flag set on the node sets that you want to retrieve

Return type

Array of NodeSet objects

Example

To make an array of NodeSet objects for all of the node sets in model m flagged with f

```
var ns = NodeSet.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the NodeSet object for a node set ID.

Arguments

Name	Type	Description
Model	Model	Model to find the node set in
number	integer	number of the node set you want the NodeSet object for

Return type

NodeSet object (or null if node set does not exist).

Example

To get the NodeSet object for node set 100 in model m

```
var ns = NodeSet.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a NodeSet property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the NodeSet.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	node set property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if NodeSet property ns.example is a parameter:

```
Options.property_parameter_names = true;
if (ns.GetParameter(ns.example) ) do_something...
Options.property_parameter_names = false;
```

To check if NodeSet property ns.example is a parameter by using the GetParameter method:

```
if (ns.ViewParameters().GetParameter(ns.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this node_set (*CONSTRAINED_NODE_SET). **Note that a carriage return is not added**. See also NodeSet.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for node_set n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the node_set. **Note that a carriage return is not added**. See also NodeSet.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for node_set n:

```
var cards = n.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last node set in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last node set in

Return type

NodeSet object (or null if there are no node sets in the model).

Example

To get the last node set in model m:

```
var ns = NodeSet.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free node set label in the model. Also see NodeSet.FirstFreeLabel(), NodeSet.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free node set label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

NodeSet label.

Example

To get the last free node set label in model m:

```
var label = NodeSet.LastFreeLabel(m);
```

Next()

Description

Returns the next node set in the model.

Arguments

No arguments

Return type

NodeSet object (or null if there are no more node sets in the model).

Example

To get the node set in model m after node set ns:

```
var ns = ns.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) node set label in the model. Also see NodeSet.FirstFreeLabel(), NodeSet.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free node set label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

NodeSet label.

Example

To get the next free node set label in model m:

```
var label = NodeSet.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a node set.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only node sets from that model can be picked. If the argument is a Flag then only node sets that are flagged with *limit* can be selected. If omitted, or null, any node sets from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

NodeSet object (or null if not picked)

Example

To pick a node set from model m giving the prompt 'Pick node set from screen':

```
var ns = NodeSet.Pick('Pick node set from screen', m);
```

Previous()

Description

Returns the previous node set in the model.

Arguments

No arguments

Return type

NodeSet object (or null if there are no more node sets in the model).

Example

To get the node set in model m before node set ns:

```
var ns = ns.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the node sets in model m, from 1000000:

```
NodeSet.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged node sets will be renumbered in
flag	Flag	Flag set on the node sets that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the node sets in model m flagged with f, from 1000000:

```
NodeSet.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select node sets using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting node sets
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only node sets from that model can be selected. If the argument is a Flag then only node sets that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any node sets can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of node sets selected or null if menu cancelled

Example

To select node sets from model m, flagging those selected with flag f, giving the prompt 'Select node sets':

```
NodeSet.Select(f, 'Select node sets', m);
```

To select node sets, flagging those selected with flag f but limiting selection to node sets flagged with flag l, giving the prompt 'Select node sets':

```
NodeSet.Select(f, 'Select node sets', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the node set.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the node set

Return type

No return value

Example

To set flag f for node set ns:

```
ns.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the node set. The node set will be sketched until you either call NodeSet.Unsketch(), NodeSet.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the node set is sketched. If omitted redraw is true. If you want to sketch several node sets and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch node set ns:

```
ns.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged node sets in the model. The node sets will be sketched until you either call NodeSet.Unsketch(), NodeSet.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged node sets will be sketched in
flag	Flag	Flag set on the node sets that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the node sets are sketched. If omitted redraw is true. If you want to sketch flagged node sets several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all node sets flagged with flag in model m:

```
NodeSet.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing node sets should be counted. If false or omitted referenced but undefined node sets will also be included in the total.

Return type

number of node sets

Example

To get the total number of node sets in model m:

```
var total = NodeSet.Total(m);
```

Unblank()

Description

Unblanks the node set

Arguments

No arguments

Return type

No return value

Example

To unblank node set ns:

```
ns.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the node sets in model m:

```
NodeSet.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged node sets will be unblanked in
flag	Flag	Flag set on the node sets that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the node sets in model m flagged with f:

```
NodeSet.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all node sets will be unset in
flag	Flag	Flag to unset on the node sets

Return type

No return value

Example

To unset the flag f on all the node sets in model m:

```
NodeSet.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the node set.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the node set is unsketched. If omitted redraw is true. If you want to unsketch several node sets and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch node set ns:

```
ns.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all node sets.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the node sets are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all node sets in model m:

```
NodeSet.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged node sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all node sets will be unsketched in
flag	Flag	Flag set on the node sets that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the node sets are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all node sets flagged with flag in model m:

```
NodeSet.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

NodeSet object.

Example

To check if NodeSet property ns.example is a parameter by using the NodeSet.GetParameter() method:

```
if (ns.ViewParameters().GetParameter(ns.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for node set. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for node set ns:

```
ns.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this node set.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for node set ns:

```
var xrefs = ns.Xrefs();
```

---

toString()

Description

Creates a string containing the node_set data in keyword format. Note that this contains the keyword header and the keyword cards. See also NodeSet.Keyword() and NodeSet.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for node set n in keyword format

```
var s = n.toString();
```

RigidBodies class

The RigidBodies class gives you access to constrained rigid bodies cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

RigidBodies properties

Name	Type	Description
exists	logical	true if constrained rigid bodies exists, false if referred to but not defined. (read only)
iflag	integer	Flag for adding coonstrained mass properties to part inertia.
include	integer	The Include file number that the constrained rigid bodies is in.
label	integer	The label the constrained rigid bodies has in PRIMER (read only)
model	integer	The Model number that the rigid body merge is in.
pidc	integer	Constrained rigid body part ID.
pidl	integer	Lead rigid body part ID.

Detailed Description

The RigidBodies class allows you to create, modify, edit and manipulate constrained rigid bodies cards. See the documentation below for more details.

Constructor

new RigidBodies(Model*[Model]*, pidl*[integer]*, pidc*[integer]*, iflag*[integer]*)

Description

Create a new RigidBodies object.

Arguments

Name	Type	Description
Model	Model	Model that constrained rigid bodies will be created in
pidl	integer	Lead rigid body part ID.
pidc	integer	Constrained rigid body part ID.
iflag	integer	Flag for adding constrained mass properties to part inertia.

Return type

RigidBodies object

Example

To create a new constrained rigid bodies in model m with lead part 5 and .constrained part 10

```
var r = new RigidBodies(m, 5, 10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a rigid body merge.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the rigid body merge

Return type

No return value

Example

To associate comment c to the rigid body merge m:

```
m.AssociateComment(c);
```

Blank()

Description

Blanks the rigid body merge

Arguments

No arguments

Return type

No return value

Example

To blank rigid body merge m:

```
m.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the rigid body merges in model m:

```
RigidBodies.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged rigid body merges will be blanked in
flag	Flag	Flag set on the rigid body merges that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the rigid body merges in model m flagged with f:

```
RigidBodies.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the rigid body merge is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if rigid body merge m is blanked:

```
if (m.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse rigid body merge m:

```
m.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the rigid body merge.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the rigid body merge

Return type

No return value

Example

To clear flag f for rigid body merge m:

```
m.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the rigid body merge.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

RigidBodies object

Example

To copy rigid body merge m into rigid body merge z:

```
var z = m.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a constrained rigid bodies definition.

Arguments

Name	Type	Description
Model	Model	Model that the constrained rigid bodies definition will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

RigidBodies object (or null if not made)

Example

To start creating a constrained rigid bodies definition in model m:

```
var r = RigidBodies.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a rigid body merge.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the rigid body merge

Return type

No return value

Example

To detach comment c from the rigid body merge m:

```
m.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit rigid body merge m:

```
m.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for rigid body merge. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for rigid body merge m:

```
m.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first rigid body merge in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first rigid body merge in

Return type

RigidBodies object (or null if there are no rigid body merges in the model).

Example

To get the first rigid body merge in model m:

```
var m = RigidBodies.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the rigid body merges in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges will be flagged in
flag	Flag	Flag to set on the rigid body merges

Return type

No return value

Example

To flag all of the rigid body merges with flag f in model m:

```
RigidBodies.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the rigid body merge is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the rigid body merge

Return type

true if flagged, false if not.

Example

To check if rigid body merge m has flag f set on it:

```
if (m.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each rigid body merge in the model.
Note that ForEach has been designed to make looping over rigid body merges as fast as possible and so has some limitations.
Firstly, a single temporary RigidBodies object is created and on each function call it is updated with the current rigid body merge data. This means that you should not try to store the RigidBodies object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new rigid body merges inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges are in
func	function	Function to call for each rigid body merge
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the rigid body merges in model m:

```
RigidBodies.ForEach(m, test);
function test(m)
{
// m is RigidBodies object
}
```

To call function test for all of the rigid body merges in model m with optional object:

```
var data = { x:0, y:0 };
RigidBodies.ForEach(m, test, data);
function test(m, extra)
{
// m is RigidBodies object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of RigidBodies objects for all of the rigid body merges in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get rigid body merges from

Return type

Array of RigidBodies objects

Example

To make an array of RigidBodies objects for all of the rigid body merges in model m

```
var m = RigidBodies.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a rigid body merge.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the rigid body merge m:

```
var comm_array = m.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of RigidBodies objects for all of the flagged rigid body merges in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get rigid body merges from
flag	Flag	Flag set on the rigid body merges that you want to retrieve

Return type

Array of RigidBodies objects

Example

To make an array of RigidBodies objects for all of the rigid body merges in model m flagged with f

```
var m = RigidBodies.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the RigidBodies object for a rigid body merge ID.

Arguments

Name	Type	Description
Model	Model	Model to find the rigid body merge in
number	integer	number of the rigid body merge you want the RigidBodies object for

Return type

RigidBodies object (or null if rigid body merge does not exist).

Example

To get the RigidBodies object for rigid body merge 100 in model m

```
var m = RigidBodies.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a RigidBodies property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the RigidBodies.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	rigid body merge property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if RigidBodies property m.example is a parameter:

```
Options.property_parameter_names = true;
if (m.GetParameter(m.example) ) do_something...
Options.property_parameter_names = false;
```

To check if RigidBodies property m.example is a parameter by using the GetParameter method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this constrained rigid bodies (*CONSTRAINED_RIGID_BODIES). **Note that a carriage return is not added**. See also RigidBodies.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for constrained rigid bodies r:

```
var key = r.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the constrained rigid bodies. **Note that a carriage return is not added**. See also RigidBodies.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for constrained rigid bodies r:

```
var cards = r.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last rigid body merge in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last rigid body merge in

Return type

RigidBodies object (or null if there are no rigid body merges in the model).

Example

To get the last rigid body merge in model m:

```
var m = RigidBodies.Last(m);
```

Next()

Description

Returns the next rigid body merge in the model.

Arguments

No arguments

Return type

RigidBodies object (or null if there are no more rigid body merges in the model).

Example

To get the rigid body merge in model m after rigid body merge m:

```
var m = m.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a rigid body merge.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only rigid body merges from that model can be picked. If the argument is a Flag then only rigid body merges that are flagged with *limit* can be selected. If omitted, or null, any rigid body merges from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

RigidBodies object (or null if not picked)

Example

To pick a rigid body merge from model m giving the prompt 'Pick rigid body merge from screen':

```
var m = RigidBodies.Pick('Pick rigid body merge from screen', m);
```

Previous()

Description

Returns the previous rigid body merge in the model.

Arguments

No arguments

Return type

RigidBodies object (or null if there are no more rigid body merges in the model).

Example

To get the rigid body merge in model m before rigid body merge m:

```
var m = m.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select rigid body merges using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting rigid body merges
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only rigid body merges from that model can be selected. If the argument is a Flag then only rigid body merges that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any rigid body merges can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of rigid body merges selected or null if menu cancelled

Example

To select rigid body merges from model m, flagging those selected with flag f, giving the prompt 'Select rigid body merges':

```
RigidBodies.Select(f, 'Select rigid body merges', m);
```

To select rigid body merges, flagging those selected with flag f but limiting selection to rigid body merges flagged with flag l, giving the prompt 'Select rigid body merges':

```
RigidBodies.Select(f, 'Select rigid body merges', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the rigid body merge.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the rigid body merge

Return type

No return value

Example

To set flag f for rigid body merge m:

```
m.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the rigid body merge. The rigid body merge will be sketched until you either call RigidBodies.Unsketch(), RigidBodies.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the rigid body merge is sketched. If omitted redraw is true. If you want to sketch several rigid body merges and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch rigid body merge m:

```
m.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged rigid body merges in the model. The rigid body merges will be sketched until you either call RigidBodies.Unsketch(), RigidBodies.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged rigid body merges will be sketched in
flag	Flag	Flag set on the rigid body merges that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the rigid body merges are sketched. If omitted redraw is true. If you want to sketch flagged rigid body merges several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all rigid body merges flagged with flag in model m:

```
RigidBodies.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing rigid body merges should be counted. If false or omitted referenced but undefined rigid body merges will also be included in the total.

Return type

number of rigid body merges

Example

To get the total number of rigid body merges in model m:

```
var total = RigidBodies.Total(m);
```

Unblank()

Description

Unblanks the rigid body merge

Arguments

No arguments

Return type

No return value

Example

To unblank rigid body merge m:

```
m.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the rigid body merges in model m:

```
RigidBodies.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged rigid body merges will be unblanked in
flag	Flag	Flag set on the rigid body merges that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the rigid body merges in model m flagged with f:

```
RigidBodies.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all rigid body merges will be unset in
flag	Flag	Flag to unset on the rigid body merges

Return type

No return value

Example

To unset the flag f on all the rigid body merges in model m:

```
RigidBodies.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the rigid body merge.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the rigid body merge is unsketched. If omitted redraw is true. If you want to unsketch several rigid body merges and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch rigid body merge m:

```
m.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all rigid body merges.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the rigid body merges are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all rigid body merges in model m:

```
RigidBodies.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged rigid body merges in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigid body merges will be unsketched in
flag	Flag	Flag set on the rigid body merges that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the rigid body merges are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all rigid body merges flagged with flag in model m:

```
RigidBodies.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

RigidBodies object.

Example

To check if RigidBodies property m.example is a parameter by using the RigidBodies.GetParameter() method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for rigid body merge. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for rigid body merge m:

```
m.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this rigid body merge.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for rigid body merge m:

```
var xrefs = m.Xrefs();
```

---

toString()

Description

Creates a string containing the constrained rigid bodies data in keyword format. Note that this contains the keyword header and the keyword cards. See also RigidBodies.Keyword() and RigidBodies.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for constrained rigid bodies r in keyword format

```
var s = r.toString();
```

Spotweld class

The Spotweld class gives you access to constrained spotweld cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Spotweld properties

Name	Type	Description
ep	real	Effective plastic strain at failure
exists	logical	true if constrained spotweld exists, false if referred to but not defined. (read only)
filtered_force	logical	true if _FILTERED_FORCE option is set, false if not
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the constrained spotweld is in.
label	integer	Constrained spotweld number
m	real	Exponent for shear spotweld force
model	integer	The Model number that the spotweld is in.
n	real	Exponent for normal spotweld force
n1	integer	Node ID
n2	integer	Node ID
nf	integer	Number of force vectors stored for filtering
sn	real	Normal force at spotweld failure
ss	real	Shear force at spotweld failure
tf	real	Failure time for nodal constraint set
tw	real	Time window for filtering
wid	integer	Constrained spotweld number (identical to label)

Detailed Description

The Spotweld class allows you to create, modify, edit and manipulate constrained spotweld cards. See the documentation below for more details.

Constructor

new Spotweld(Model*[Model]*, n1*[integer]*, n2*[integer]*, label (optional)*[integer]*)

Description

Create a new Spotweld object.

Arguments

Name	Type	Description
Model	Model	Model that constrained spotweld will be created in
n1	integer	Node ID 1
n2	integer	Node ID 2
label (optional)	integer	Constrained spotweld number

Return type

Spotweld object

Example

To create a new constrained spotweld 500 in model m between nodes 10 and 11

```
var s = new Spotweld(m, 10, 11, 500);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a spotweld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the spotweld

Return type

No return value

Example

To associate comment c to the spotweld s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the spotweld

Arguments

No arguments

Return type

No return value

Example

To blank spotweld s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the spotwelds in model m:

```
Spotweld.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged spotwelds will be blanked in
flag	Flag	Flag set on the spotwelds that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the spotwelds in model m flagged with f:

```
Spotweld.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the spotweld is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if spotweld s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse spotweld s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the spotweld.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the spotweld

Return type

No return value

Example

To clear flag f for spotweld s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the spotweld.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Spotweld object

Example

To copy spotweld s into spotweld z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a spotweld.

Arguments

Name	Type	Description
Model	Model	Model that the spotweld will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Spotweld object (or null if not made)

Example

To start creating a spotweld in model s:

```
var s = Spotweld.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a spotweld.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the spotweld

Return type

No return value

Example

To detach comment c from the spotweld s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit spotweld s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for spotweld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for spotweld s:

```
s.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first spotweld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first spotweld in

Return type

Spotweld object (or null if there are no spotwelds in the model).

Example

To get the first spotweld in model m:

```
var s = Spotweld.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free spotweld label in the model. Also see Spotweld.LastFreeLabel(), Spotweld.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free spotweld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Spotweld label.

Example

To get the first free spotweld label in model m:

```
var label = Spotweld.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the spotwelds in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds will be flagged in
flag	Flag	Flag to set on the spotwelds

Return type

No return value

Example

To flag all of the spotwelds with flag f in model m:

```
Spotweld.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the spotweld is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the spotweld

Return type

true if flagged, false if not.

Example

To check if spotweld s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each spotweld in the model.
Note that ForEach has been designed to make looping over spotwelds as fast as possible and so has some limitations.
Firstly, a single temporary Spotweld object is created and on each function call it is updated with the current spotweld data. This means that you should not try to store the Spotweld object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new spotwelds inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds are in
func	function	Function to call for each spotweld
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the spotwelds in model m:

```
Spotweld.ForEach(m, test);
function test(s)
{
// s is Spotweld object
}
```

To call function test for all of the spotwelds in model m with optional object:

```
var data = { x:0, y:0 };
Spotweld.ForEach(m, test, data);
function test(s, extra)
{
// s is Spotweld object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Spotweld objects for all of the spotwelds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get spotwelds from

Return type

Array of Spotweld objects

Example

To make an array of Spotweld objects for all of the spotwelds in model m

```
var s = Spotweld.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a spotweld.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the spotweld s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Spotweld objects for all of the flagged spotwelds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get spotwelds from
flag	Flag	Flag set on the spotwelds that you want to retrieve

Return type

Array of Spotweld objects

Example

To make an array of Spotweld objects for all of the spotwelds in model m flagged with f

```
var s = Spotweld.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Spotweld object for a spotweld ID.

Arguments

Name	Type	Description
Model	Model	Model to find the spotweld in
number	integer	number of the spotweld you want the Spotweld object for

Return type

Spotweld object (or null if spotweld does not exist).

Example

To get the Spotweld object for spotweld 100 in model m

```
var s = Spotweld.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Spotweld property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Spotweld.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	spotweld property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Spotweld property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Spotweld property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this spotweld (*CONSTRAINED_SPOTWELD). **Note that a carriage return is not added**. See also Spotweld.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for spotweld s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the spotweld. **Note that a carriage return is not added**. See also Spotweld.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for spotweld s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last spotweld in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last spotweld in

Return type

Spotweld object (or null if there are no spotwelds in the model).

Example

To get the last spotweld in model m:

```
var s = Spotweld.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free spotweld label in the model. Also see Spotweld.FirstFreeLabel(), Spotweld.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free spotweld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Spotweld label.

Example

To get the last free spotweld label in model m:

```
var label = Spotweld.LastFreeLabel(m);
```

Next()

Description

Returns the next spotweld in the model.

Arguments

No arguments

Return type

Spotweld object (or null if there are no more spotwelds in the model).

Example

To get the spotweld in model m after spotweld s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) spotweld label in the model. Also see Spotweld.FirstFreeLabel(), Spotweld.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free spotweld label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Spotweld label.

Example

To get the next free spotweld label in model m:

```
var label = Spotweld.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a spotweld.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only spotwelds from that model can be picked. If the argument is a Flag then only spotwelds that are flagged with *limit* can be selected. If omitted, or null, any spotwelds from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Spotweld object (or null if not picked)

Example

To pick a spotweld from model m giving the prompt 'Pick spotweld from screen':

```
var s = Spotweld.Pick('Pick spotweld from screen', m);
```

Previous()

Description

Returns the previous spotweld in the model.

Arguments

No arguments

Return type

Spotweld object (or null if there are no more spotwelds in the model).

Example

To get the spotweld in model m before spotweld s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the spotwelds in model m, from 1000000:

```
Spotweld.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged spotwelds will be renumbered in
flag	Flag	Flag set on the spotwelds that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the spotwelds in model m flagged with f, from 1000000:

```
Spotweld.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select spotwelds using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting spotwelds
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only spotwelds from that model can be selected. If the argument is a Flag then only spotwelds that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any spotwelds can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of spotwelds selected or null if menu cancelled

Example

To select spotwelds from model m, flagging those selected with flag f, giving the prompt 'Select spotwelds':

```
Spotweld.Select(f, 'Select spotwelds', m);
```

To select spotwelds, flagging those selected with flag f but limiting selection to spotwelds flagged with flag l, giving the prompt 'Select spotwelds':

```
Spotweld.Select(f, 'Select spotwelds', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the spotweld.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the spotweld

Return type

No return value

Example

To set flag f for spotweld s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the spotweld. The spotweld will be sketched until you either call Spotweld.Unsketch(), Spotweld.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the spotweld is sketched. If omitted redraw is true. If you want to sketch several spotwelds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch spotweld s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged spotwelds in the model. The spotwelds will be sketched until you either call Spotweld.Unsketch(), Spotweld.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged spotwelds will be sketched in
flag	Flag	Flag set on the spotwelds that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the spotwelds are sketched. If omitted redraw is true. If you want to sketch flagged spotwelds several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all spotwelds flagged with flag in model m:

```
Spotweld.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing spotwelds should be counted. If false or omitted referenced but undefined spotwelds will also be included in the total.

Return type

number of spotwelds

Example

To get the total number of spotwelds in model m:

```
var total = Spotweld.Total(m);
```

Unblank()

Description

Unblanks the spotweld

Arguments

No arguments

Return type

No return value

Example

To unblank spotweld s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the spotwelds in the model.

Arguments

Name	Type	Description

Model	Model	Model that all spotwelds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the spotwelds in model m:

```
Spotweld.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged spotwelds will be unblanked in
flag	Flag	Flag set on the spotwelds that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the spotwelds in model m flagged with f:

```
Spotweld.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all spotwelds will be unset in
flag	Flag	Flag to unset on the spotwelds

Return type

No return value

Example

To unset the flag f on all the spotwelds in model m:

```
Spotweld.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the spotweld.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the spotweld is unsketched. If omitted redraw is true. If you want to unsketch several spotwelds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch spotweld s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all spotwelds.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the spotwelds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all spotwelds in model m:

```
Spotweld.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged spotwelds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all spotwelds will be unsketched in
flag	Flag	Flag set on the spotwelds that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the spotwelds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all spotwelds flagged with flag in model m:

```
Spotweld.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Spotweld object.

Example

To check if Spotweld property s.example is a parameter by using the Spotweld.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for spotweld. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for spotweld s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this spotweld.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for spotweld s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the spotweld data in keyword format. Note that this contains the keyword header and the keyword cards. See also Spotweld.Keyword() and Spotweld.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for spotweld s in keyword format

```
var str = s.toString();
```

Spr2 class

The Spr2 class gives you access to constrained spr2 cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Spr2 constants

Constants for Flags for Interpolation

Name	Description

Spr2.INVERSE	Property INTP value EQ.2.0: Inverse distance weighting.
Spr2.LINEAR	Property INTP value EQ.0.0: Linear (default).
Spr2.UNIFORM	Property INTP value EQ.1.0: Uniform.

Spr2 properties

Name	Type	Description
alpha1	real	Dimensionless parameter scaling the effective displacement.
alpha2	real	Dimensionless parameter scaling the effective displacement.
alpha3	real	Dimensionless parameter scaling the effective displacement. (GT.0: incremental update (default), LT.0: total update (recommended))
d	real	Rivet diameter.
dens	real	Rivet density (necessary for time step calculation).
dn	real	Failure displacement in normal direction.
dt	real	Failure displacement in tangential direction.
exists	logical	true if constrained spr2 exists, false if referred to but not defined. (read only)
expn	real	Exponent value for load function in normal direction.
expt	real	Exponent value for load function in tangential direction.
fn	real	Rivet strength in tension (pull-out) or (if -ve) label for UPID
ft	real	Rivet strength in pure shear.
include	integer	The Include file number that the constrained spr2 is in.
intp	real	Flag for interpolation. Values can be Spr2.LINEAR, Spr2.UNIFORM or Spr2.INVERSE.
lpid	integer	Lower Sheet Part ID.
model	integer	The Model number that the spr2 is in.
nsid	integer	Node Set ID of rivet location nodes.
pidvb	integer	Part ID for visualization beams representing SPR2 in postprocessing.
thick	real	Total thickness of master and slave sheet.
upid	integer	Upper Sheet Part ID.
xin	real	Fraction of failure displacement at maximum normal force.
xit	real	Fraction of failure displacement at maximum tangential force.
xpid1	integer	Extra Part ID 1 for multi-sheet connection.
xpid2	integer	Extra Part ID 2 for multi-sheet connection.
xpid3	integer	Extra Part ID 3 for multi-sheet connection.
xpid4	integer	Extra Part ID 4 for multi-sheet connection.

Detailed Description

The Spr2 class allows you to create, modify, edit and manipulate constrained spr2 cards. See the documentation below for more details.

Constructor

new Spr2(Model*[Model]*, upid*[integer]*, lpid*[integer]*, nsid*[integer]*)

Description

Create a new Spr2 object.

Arguments

Name	Type	Description
Model	Model	Model that constrained spr2 will be created in
upid	integer	Upper Sheet Part ID.
lpid	integer	Lower Sheet Part ID
nsid	integer	Node Set ID of rivet location nodes.

Return type

Spr2 object

Example

To create a new constrained spr2 in model m with master sheet 100, slave sheet 200 and rivet node set 100

```
var s = new Spr2(m, 100, 200, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a spr2.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the spr2

Return type

No return value

Example

To associate comment c to the spr2 s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the spr2

Arguments

No arguments

Return type

No return value

Example

To blank spr2 s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model that all spr2s will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the spr2s in model m:

```
Spr2.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged spr2s will be blanked in
flag	Flag	Flag set on the spr2s that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the spr2s in model m flagged with f:

```
Spr2.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the spr2 is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if spr2 s is blanked:

```
if (s.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the spr2.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the spr2

Return type

No return value

Example

To clear flag f for spr2 s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the spr2.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Spr2 object

Example

To copy spr2 s into spr2 z:

```
var z = s.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a spr2.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the spr2

Return type

No return value

Example

To detach comment c from the spr2 s:

```
s.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for spr2. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for spr2 s:

```
s.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first spr2 in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first spr2 in

Return type

Spr2 object (or null if there are no spr2s in the model).

Example

To get the first spr2 in model m:

```
var s = Spr2.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the spr2s in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all spr2s will be flagged in
flag	Flag	Flag to set on the spr2s

Return type

No return value

Example

To flag all of the spr2s with flag f in model m:

```
Spr2.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the spr2 is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the spr2

Return type

true if flagged, false if not.

Example

To check if spr2 s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each spr2 in the model.
Note that ForEach has been designed to make looping over spr2s as fast as possible and so has some limitations. Firstly, a single temporary Spr2 object is created and on each function call it is updated with the current spr2 data. This means that you should not try to store the Spr2 object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new spr2s inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all spr2s are in
func	function	Function to call for each spr2
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the spr2s in model m:

```
Spr2.ForEach(m, test);
function test(s)
{
// s is Spr2 object
}
```

To call function test for all of the spr2s in model m with optional object:

```
var data = { x:0, y:0 };
Spr2.ForEach(m, test, data);
function test(s, extra)
{
// s is Spr2 object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Spr2 objects for all of the spr2s in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get spr2s from

Return type

Array of Spr2 objects

Example

To make an array of Spr2 objects for all of the spr2s in model m

```
var s = Spr2.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a spr2.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the spr2 s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Spr2 objects for all of the flagged spr2s in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get spr2s from
flag	Flag	Flag set on the spr2s that you want to retrieve

Return type

Array of Spr2 objects

Example

To make an array of Spr2 objects for all of the spr2s in model m flagged with f

```
var s = Spr2.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Spr2 object for a spr2 ID.

Arguments

Name	Type	Description
Model	Model	Model to find the spr2 in
number	integer	number of the spr2 you want the Spr2 object for

Return type

Spr2 object (or null if spr2 does not exist).

Example

To get the Spr2 object for spr2 100 in model m

```
var s = Spr2.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Spr2 property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Spr2.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	spr2 property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Spr2 property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Spr2 property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this spr2 (*CONSTRAINED_SPR2). **Note that a carriage return is not added**. See also Spr2.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for spr2 s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the spr2. **Note that a carriage return is not added**. See also Spr2.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for spr2 s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last spr2 in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last spr2 in

Return type

Spr2 object (or null if there are no spr2s in the model).

Example

To get the last spr2 in model m:

```
var s = Spr2.Last(m);
```

Next()

Description

Returns the next spr2 in the model.

Arguments

No arguments

Return type

Spr2 object (or null if there are no more spr2s in the model).

Example

To get the spr2 in model m after spr2 s:

```
var s = s.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a spr2.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only spr2s from that model can be picked. If the argument is a Flag then only spr2s that are flagged with *limit* can be selected. If omitted, or null, any spr2s from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Spr2 object (or null if not picked)

Example

To pick a spr2 from model m giving the prompt 'Pick spr2 from screen':

```
var s = Spr2.Pick('Pick spr2 from screen', m);
```

Previous()

Description

Returns the previous spr2 in the model.

Arguments

No arguments

Return type

Spr2 object (or null if there are no more spr2s in the model).

Example

To get the spr2 in model m before spr2 s:

```
var s = s.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select spr2s using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting spr2s
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only spr2s from that model can be selected. If the argument is a Flag then only spr2s that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any spr2s can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of spr2s selected or null if menu cancelled

Example

To select spr2s from model m, flagging those selected with flag f, giving the prompt 'Select spr2s':

```
Spr2.Select(f, 'Select spr2s', m);
```

To select spr2s, flagging those selected with flag f but limiting selection to spr2s flagged with flag l, giving the prompt 'Select spr2s':

```
Spr2.Select(f, 'Select spr2s', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the spr2.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the spr2

Return type

No return value

Example

To set flag f for spr2 s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the spr2. The spr2 will be sketched until you either call Spr2.Unsketch(), Spr2.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the spr2 is sketched. If omitted redraw is true. If you want to sketch several spr2s and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch spr2 s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged spr2s in the model. The spr2s will be sketched until you either call Spr2.Unsketch(), Spr2.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged spr2s will be sketched in
flag	Flag	Flag set on the spr2s that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the spr2s are sketched. If omitted redraw is true. If you want to sketch flagged spr2s several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all spr2s flagged with flag in model m:

```
Spr2.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing spr2s should be counted. If false or omitted referenced but undefined spr2s will also be included in the total.

Return type

number of spr2s

Example

To get the total number of spr2s in model m:

```
var total = Spr2.Total(m);
```

Unblank()

Description

Unblanks the spr2

Arguments

No arguments

Return type

No return value

Example

To unblank spr2 s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the spr2s in the model.

Arguments

Name	Type	Description

Model	Model	Model that all spr2s will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the spr2s in model m:

```
Spr2.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged spr2s will be unblanked in
flag	Flag	Flag set on the spr2s that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the spr2s in model m flagged with f:

```
Spr2.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all spr2s will be unset in
flag	Flag	Flag to unset on the spr2s

Return type

No return value

Example

To unset the flag f on all the spr2s in model m:

```
Spr2.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the spr2.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the spr2 is unsketched. If omitted redraw is true. If you want to unsketch several spr2s and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch spr2 s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all spr2s.

Arguments

Name	Type	Description
Model	Model	Model that all spr2s will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the spr2s are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all spr2s in model m:

```
Spr2.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged spr2s in the model.

Arguments

Name	Type	Description
Model	Model	Model that all spr2s will be unsketched in
flag	Flag	Flag set on the spr2s that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the spr2s are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all spr2s flagged with flag in model m:

```
Spr2.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Spr2 object.

Example

To check if Spr2 property s.example is a parameter by using the Spr2.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for spr2. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for spr2 s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this spr2.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for spr2 s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the spr2 data in keyword format. Note that this contains the keyword header and the keyword cards. See also Spr2.Keyword() and Spr2.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for spr2 s in keyword format

```
var str = s.toString();
```

TieBreak class

The TieBreak class gives you access to constrained Tie-Break cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

TieBreak properties

Name	Type	Description
eppf	real	Plastic strain at failure.

exists	logical	true if constrained tie-break exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the constrained tie-break is in.
model	integer	The Model number that the tie-break is in.
nsid1	integer	First Node Set ID.
nsid2	integer	Second Node Set ID.

Detailed Description

The TieBreak class allows you to create, modify, edit and manipulate constrained tie-break cards. See the documentation below for more details.

Constructor

new TieBreak(Model*[Model]*, nsid1*[integer]*, nsid2*[integer]*, eppf (optional)*[real]*)

Description

Create a new TieBreak object.

Arguments

Name	Type	Description
Model	Model	Model that constrained tie-break will be created in
nsid1	integer	First Node Set ID.
nsid2	integer	Second Node Set ID.
eppf (optional)	real	Plastic strain at failure.

Return type

TieBreak object

Example

To create a new constrained tie-break in model m with first node set 100, second node set 200

```
var tb = new TieBreak(m, 100, 200);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a tie-break.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the tie-break

Return type

No return value

Example

To associate comment c to the tie-break tb:

```
tb.AssociateComment(c);
```

Blank()

Description

Blanks the tie-break

Arguments

No arguments

Return type

No return value

Example

To blank tie-break tb:

```
tb.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the tie-breaks in model m:

```
TieBreak.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged tie-breaks will be blanked in
flag	Flag	Flag set on the tie-breaks that you want to blank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the tie-breaks in model m flagged with f:

```
TieBreak.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the tie-break is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if tie-break tb is blanked:

```
if (tb.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the tie-break.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the tie-break

Return type

No return value

Example

To clear flag f for tie-break tb:

```
tb.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the tie-break.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

TieBreak object

Example

To copy tie-break tb into tie-break z:

```
var z = tb.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a tie-break.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the tie-break

Return type

No return value

Example

To detach comment c from the tie-break tb:

```
tb.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for tie-break. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for tie-break tb:

```
tb.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first tie-break in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first tie-break in

Return type

TieBreak object (or null if there are no tie-breaks in the model).

Example

To get the first tie-break in model m:

```
var tb = TieBreak.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the tie-breaks in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks will be flagged in
flag	Flag	Flag to set on the tie-breaks

Return type

No return value

Example

To flag all of the tie-breaks with flag f in model m:

```
TieBreak.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the tie-break is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the tie-break

Return type

true if flagged, false if not.

Example

To check if tie-break tb has flag f set on it:

```
if (tb.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each tie-break in the model.
Note that ForEach has been designed to make looping over tie-breaks as fast as possible and so has some limitations.
Firstly, a single temporary TieBreak object is created and on each function call it is updated with the current tie-break data. This means that you should not try to store the TieBreak object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new tie-breaks inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks are in
func	function	Function to call for each tie-break
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the tie-breaks in model m:

```
TieBreak.ForEach(m, test);
function test(tb)
{
// tb is TieBreak object
}
```

To call function test for all of the tie-breaks in model m with optional object:

```
var data = { x:0, y:0 };
TieBreak.ForEach(m, test, data);
function test(tb, extra)
{
// tb is TieBreak object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of TieBreak objects for all of the tie-breaks in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get tie-breaks from

Return type

Array of TieBreak objects

Example

To make an array of TieBreak objects for all of the tie-breaks in model m

```
var tb = TieBreak.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a tie-break.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the tie-break tb:

```
var comm_array = tb.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of TieBreak objects for all of the flagged tie-breaks in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get tie-breaks from
flag	Flag	Flag set on the tie-breaks that you want to retrieve

Return type

Array of TieBreak objects

Example

To make an array of TieBreak objects for all of the tie-breaks in model m flagged with f

```
var tb = TieBreak.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the TieBreak object for a tie-break ID.

Arguments

Name	Type	Description
Model	Model	Model to find the tie-break in
number	integer	number of the tie-break you want the TieBreak object for

Return type

TieBreak object (or null if tie-break does not exist).

Example

To get the TieBreak object for tie-break 100 in model m

```
var tb = TieBreak.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a TieBreak property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the TieBreak.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	tie-break property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if TieBreak property tb.example is a parameter:

```
Options.property_parameter_names = true;
if (tb.GetParameter(tb.example) ) do_something...
Options.property_parameter_names = false;
```

To check if TieBreak property tb.example is a parameter by using the GetParameter method:

```
if (tb.ViewParameters().GetParameter(tb.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this tie-break (**CONSTRAINED_TIE_BREAK). **Note that a carriage return is not added**. See also TieBreak.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for tie-break tb:

```
var key = tb.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the tie-break. **Note that a carriage return is not added**. See also TieBreak.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for tie-break tb:

```
var cards = tb.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last tie-break in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last tie-break in

Return type

TieBreak object (or null if there are no tie-breaks in the model).

Example

To get the last tie-break in model m:

```
var tb = TieBreak.Last(m);
```

Next()

Description

Returns the next tie-break in the model.

Arguments

No arguments

Return type

TieBreak object (or null if there are no more tie-breaks in the model).

Example

To get the tie-break in model m after tie-break tb:

```
var tb = tb.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a tie-break.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only tie-breaks from that model can be picked. If the argument is a Flag then only tie-breaks that are flagged with *limit* can be selected. If omitted, or null, any tie-breaks from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

TieBreak object (or null if not picked)

Example

To pick a tie-break from model m giving the prompt 'Pick tie-break from screen':

```
var tb = TieBreak.Pick('Pick tie-break from screen', m);
```

Previous()

Description

Returns the previous tie-break in the model.

Arguments

No arguments

Return type

TieBreak object (or null if there are no more tie-breaks in the model).

Example

To get the tie-break in model m before tie-break tb:

```
var tb = tb.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select tie-breaks using standard PRIMER object menus.

Arguments

Name	Type	Description

flag	Flag	Flag to use when selecting tie-breaks
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only tie-breaks from that model can be selected. If the argument is a Flag then only tie-breaks that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any tie-breaks can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of tie-breaks selected or null if menu cancelled

Example

To select tie-breaks from model m, flagging those selected with flag f, giving the prompt 'Select tie-breaks':

```
TieBreak.Select(f, 'Select tie-breaks', m);
```

To select tie-breaks, flagging those selected with flag f but limiting selection to tie-breaks flagged with flag l, giving the prompt 'Select tie-breaks':

```
TieBreak.Select(f, 'Select tie-breaks', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the tie-break.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the tie-break

Return type

No return value

Example

To set flag f for tie-break tb:

```
tb.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the tie-break. The tie-break will be sketched until you either call TieBreak.Unsketch(), TieBreak.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the tie-break is sketched. If omitted redraw is true. If you want to sketch several tie-breaks and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch tie-break tb:

```
tb.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged tie-breaks in the model. The tie-breaks will be sketched until you either call TieBreak.Unsketch(), TieBreak.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged tie-breaks will be sketched in
flag	Flag	Flag set on the tie-breaks that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the tie-breaks are sketched. If omitted redraw is true. If you want to sketch flagged tie-breaks several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all tie-breaks flagged with flag in model m:

```
TieBreak.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing tie-breaks should be counted. If false or omitted referenced but undefined tie-breaks will also be included in the total.

Return type

number of tie-breaks

Example

To get the total number of tie-breaks in model m:

```
var total = TieBreak.Total(m);
```

Unblank()

Description

Unblanks the tie-break

Arguments

No arguments

Return type

No return value

Example

To unblank tie-break tb:

```
tb.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the tie-breaks in model m:

```
TieBreak.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged tie-breaks will be unblanked in
flag	Flag	Flag set on the tie-breaks that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the tie-breaks in model m flagged with f:

```
TieBreak.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all tie-breaks will be unset in
flag	Flag	Flag to unset on the tie-breaks

Return type

No return value

Example

To unset the flag f on all the tie-breaks in model m:

```
TieBreak.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the tie-break.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the tie-break is unsketched. If omitted redraw is true. If you want to unsketch several tie-breaks and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch tie-break tb:

```
tb.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all tie-breaks.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the tie-breaks are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all tie-breaks in model m:

```
TieBreak.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged tie-breaks in the model.

Arguments

Name	Type	Description
Model	Model	Model that all tie-breaks will be unsketched in
flag	Flag	Flag set on the tie-breaks that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the tie-breaks are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all tie-breaks flagged with flag in model m:

```
TieBreak.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

TieBreak object.

Example

To check if TieBreak property tb.example is a parameter by using the TieBreak.GetParameter() method:

```
if (tb.ViewParameters().GetParameter(tb.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for tie-break. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for tie-break tb:

```
tb.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this tie-break.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for tie-break tb:

```
var xrefs = tb.Xrefs();
```

---

toString()

Description

Creates a string containing the tie-break data in keyword format. Note that this contains the keyword header and the keyword cards. See also TieBreak.Keyword() and TieBreak.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for tie-break tb in keyword format

```
var str = tb.toString();
```

Contact class

The Contact class gives you access to define contact cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Constrained(connection (optional)*[boolean]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- FindInteractions() [deprecated]
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Interactions(type (optional)*[constant]*)
- Keyword()
- KeywordCards()
- Next()
- PenCheck(flag*[Flag]*, eflag*[integer]*)
- PenCheckEdit(modal (optional)*[boolean]*, check_mode (optional)*[constant]*, mpp_threshold (optional)*[real]*, report_crossed_3d_elems (optional)*[boolean]*, contact_penchk_dup_shells (optional)*[constant]*)
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)

- StatusCheck()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Contact constants

Name	Description
Contact.CROSSED_EDGES	Return crossed edges in Contact.Interactions()
Contact.MPP_MODE	MPP penetration check mode
Contact.PENETRATIONS	Return penetrations in Contact.Interactions()
Contact.SMP_MODE	SMP penetration check mode

Constants for Contact _OFFSET types

Name	Description
Contact.BEAM_OFFSET	Adds _BEAM_OFFSET option
Contact.CONSTR_OFFSET	Adds _CONSTRAINED_OFFSET option
Contact.NO_OFFSET	No offset option added.
Contact.SIMPLE_OFFSET	Adds _OFFSET option

Constants for Contact penetration check_mode types

Name	Description
Contact.MPP_METHOD	Launches the penetration edit panel with the MPP methodology turned on
Contact.SMP_METHOD	Launches the penetration edit panel with the SMP methodology turned on

Constants for Contact penetration contact_penchk_dup_shells types

Name	Description
Contact.SHELL_AUTO	Launches the penetration edit panel with Automatic shell treatment of duplicate shells.
Contact.SHELL_THICK	Launches the penetration edit panel with the thickest always option for duplicate shells.
Contact.SHELL_THIN	Launches the penetration edit panel with the thinnest always option for duplicate shells.

Contact properties

Name	Type	Description
bt	real	Contact birth time
check_mode	integer	Checking mode on the pen check edit panel. (Can be Contact.MPP_METHOD, Contact.MPP_METHOD or Contact.SMP_METHOD)
cid	integer	Contact number (identical to label.
contact_penchk_dup_shells	integer	Shell treatment on the pen check edit panel. (Can be Contact.SHELL_AUTO, Contact.SHELL_AUTO or Contact.SHELL_THIN or Contact.SHELL_THICK)
dc	real	Exponential decay coeff

dt	real	Contact death time
exists	logical	true if contact exists, false if referred to but not defined. (read only)
fd	real	Dynamic coeff of friction
fs	real	Static coeff of friction
fsf	real	Coulomb friction scale factor
heading	string	Contact heading
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the contact is in.
label	integer	Contact number.
mboxid	integer	Master box id
model	integer	The Model number that the contact is in.
mortar	logical	_MORTAR keyword option - true if set, false if not.
mpr	integer	Master side printout flag
msid	integer	Master set id
mst	real	Optional master side shell thickness
mstyp	integer	Master set type
offset_flag	integer	_OFFSET option. (Can be Contact.NO_OFFSET, Contact.SIMPLE_OFFSET, Contact.CONSTR_OFFSET or Contact.BEAM_OFFSET)
penchk	integer	Penetration search flag
sboxid	integer	Slave box id
sfm	real	Scale factor on master penalty stiffness
sfmt	real	Scale factor on true master shell thickness
sfs	real	Scale factor on slave penalty stiffness
sfst	real	Scale factor on true slave shell thickness
spr	integer	Slave side printout flag
ssid	integer	Slave set id
sst	real	Optional slave side shell thickness
sstyp	integer	Slave set type
type	string	Contact type ("AUTOMATIC_GENERAL", "SINGLE_SURFACE" etc).
vc	real	Coeff for viscous friction
vdc	real	Visous damping coefficient
vsf	real	Viscous friction scale factor

Properties for COMPOSITE/LUBRICATION options

Name	Type	Description
cideta	integer	Curve ID for the viscosity
cidmu	integer	Curve ID for the coefficient of friction
d_comp	real	Composite film thickness
tfail	real	Tensile traction for failure

Properties for CONSTRAINT options

Name	Type	Description
kpf	real	Kinematic partition factor

Properties for CONTRACTION_JOINT options

Name	Type	Description
alpha	real	Key amplitude parameter A
beta	real	Key amplitude parameter B
mtcj	integer	The method option for the gap function
tsvx	real	X component of the free sliding direction T
tsvy	real	Y component of the free sliding direction T
tsvz	real	Z component of the free sliding direction T

Properties for DRAWBEAD options

Name	Type	Description
dbdth	real	Draw bead depth
dbpid	integer	optional Part ID
dfscl	real	Scale factor on lcidrf
eloff	integer	optional element id offset
ending	real	Parameter to define the length of the bead
epm	real	Maximum strain the blank will experience when it passes the bead
epscale	real	Scale factor to weaken the stress-strain curve
lceps	integer	Loadcurve ID for plastic strain vs. parametric coord
lceps2	integer	Loadcurve ID for plastic strain vs. parametric coord (elements moved > offset)
lcidnf	integer	Loadcurve ID for Normal force per unit length
lcidrf	integer	Loadcurve ID for Force due to bending per unit length
numint	integer	#int points along drawbead
offset	real	distance offset
tscale	integer	

Properties for ERODING options

Name	Type	Description
erosop	integer	Erosion/interior node option
iadj	integer	Adjacent matl treatment for solids
isym	integer	Symmetry plane option

Properties for INTERFERENCE options

Name	Type	Description
lcid1	integer	Loadcurve ID for Dyn rel stiffness
lcid2	integer	Loadcurve ID for Transient stiffness

Properties for RIGID options

Name	Type	Description
fcm	integer	Force calculation method
lcid	integer	Loadcurve ID for Force vs penetration curve
us	real	Optional unloading stiffness

Properties for THERMAL options

Name	Type	Description
a	integer	Loadcurve ID for a
algo	integer	contact algorithm
b	integer	Loadcurve ID for b
bc_flg	integer	boundary condition flag
c	integer	Loadcurve ID for c
d	integer	Loadcurve ID for d
formula	integer	formula id
frad	real	Radiation conductance across gap
ftoslv	real	Fraction of sliding friction energy partitioned to slave surface
h0	real	Heat transfer coefficient
k	real	Conductivity of gap fluid
lcfdt	integer	Loadcurve ID for dynamic friction vs. temp
lcfst	integer	Loadcurve ID for static friction vs. temp
lch	integer	Loadcurve ID for lch
lmax	real	Max size for thermal contact
lmin	real	Critical gap size
thermal	logical	If _THERMAL option is set. Can be true or false

Properties for TIEBREAK options

Name	Type	Description
cn	real	Normal stiffness
ct2cn	real	Ratio of tangential stiffness to normal stiffness
eraten	real	Normal energy release rate used in damage calculation
erates	real	Shear energy release rate used in damage calculation
mes	real	Shear force exponent
nen	real	Normal force exponent
nfls	real	Normal failure stress
option	integer	Response option
param	real	Critical distance
sfls	real	Shear failure stress
tblcid	integer	Loadcurve ID for stress vs gap post failure

thkoff	integer	flag for thickness offset

Properties for TIEBREAK_USER options

Name	Type	Description
cn	real	Normal stiffness
ct2cn	real	Ratio of tangential to normal stiff
nhv	integer	Number of history variables
offset	integer	Flag for offset treatment. This is only valid for *CONTACT_AUTOMATIC(_ONE_WAY)_ SURFACE_TO_SURFACE_TIEBREAK_USER and should not be confused with the 'offset' property for other contact types.
option	integer	User tiebreak type
up1	real	User parameter
up10	real	User parameter
up11	real	User parameter
up12	real	User parameter
up13	real	User parameter
up14	real	User parameter
up15	real	User parameter
up16	real	User parameter
up2	real	User parameter
up3	real	User parameter
up4	real	User parameter
up5	real	User parameter
up6	real	User parameter
up7	real	User parameter
up8	real	User parameter
up9	real	User parameter

Properties for TIED_WELD options

Name	Type	Description
close	real	Surface closeness parameter
hclose	real	Thermal contact conductivity
nmhis	integer	Number of material history variables
nmtwh	integer	Number of master tied weld history variables
nstwh	integer	Number of slave tied weld history variables
ntprm	integer	Number of user tied weld parameters
temp	real	Minimum temperature required.

Properties for _MPP option

Name	Type	Description

bucket	integer	Bucket sorting frequency
chksegs	integer	Special check for inverted elements
cparm8	integer	Exclude beam to beam contact flag
grpable	integer	Experimental contact algorithm
inititer	integer	Number of iterations for initial penetration checking
lcbucket	integer	Bucket sorting frequency loadcurve ID
mpp	logical	true if _MPP option is set, false if not
ns2track	integer	Number of segments to track per slave node
parmax	real	The parametric extension distance for contact segments
pensf	real	Ignore penetration scale factor

Properties for optional card A

Name	Type	Description
bsort	integer	Loadcurve for #cycles between bucket sorts
depth	integer	Loadcurve for search depth in automatic contact
frcfrq	integer	#cycles between penalty force updates
lcidab	integer	Loadcurve ID for airbag thickness vs time
maxpar	real	Max parametric coord overlap
sbopt	real	segment based contact option
sofscl	real	Soft constraint scale factor
soft	integer	Soft constraint flag

Properties for optional card B

Name	Type	Description
i2d3d	integer	Segment searching option
isym	integer	Symmetry plane option
penmax	real	Max pen distance for "old" types 3, 5, 10
shlthk	integer	Thickness consideration flag
sldstf	real	Optional solid stiffness
sldthk	real	Optional solid thickness
snlog	integer	Shooting node logic flag
thkopt	integer	Thickness option for "old" types 3, 5, 10

Properties for optional card C

Name	Type	Description
cid_rcf	integer	Coordinate system ID to output rcforc force resultants and ncforc data in a local system
dprfac	real	Depth of penetration reduction factor
dtstif	real	Timestep used in stiffness calc
flangl	real	Angle tolerance in radians for feature lines option in smooth contact

igap	integer	Implicit convergence flag
ignore	integer	Ignore initial pens in automatic types

Properties for optional card D

Name	Type	Description
dnlscl	real	Distance for nonlinear force scaling
dtpchk	real	Time interval between penetration reports
fnlscl	real	Scale factor for nonlinear force scaling
q2tri	integer	Split quads into 2 trias
sfnbr	real	Scale factor for neighbour segment contact
shledg	integer	Edge shape for shells when measuring penetration
tcso	integer	Segment treatment only flag
tiedid	integer	Incremental displacement update for tied contacts

Properties for optional card E

Name	Type	Description
cparm8smp	integer	Spotweld beam flag for SMP
fricsf	real	Scale factor for frictional stiffness
ftorq	integer	Beam torsional force computation flag
icor	integer	coefficient of restitution expressed as a percentage
ipback	integer	Create backup penalty tied contact
region	integer	Region to limit contact volume
sharec	integer	Shared constraint flag
srnde	integer	Flag for non-extended exterior shell edges

Properties for optional card F

Name	Type	Description
dbinr	integer	2dbinr - Flag to include 2d belt elements in contact (note properties cannot start with a number, so 2 has been removed).
fstol	real	Tolerance for determining flat segments.
ignroff	integer	Flag to ignore the thickness offset for shells in the calculation of the shell contact penetration depth.
pstiff	integer	Flag to choose the method for calculating the penalty stiffness
ssftyp	integer	Flag to determine how the SSF option on *PART_CONTACT behaves when SOFT = 2 on optional card A
swtpr	integer	Flag to use tapered shell contact segments adjacent to segments that are thinned by the SPOTHIN option on *CONTROL_CONTACT
tetfac	real	Scale factor for the computed volume of tetrahedral solid elements for the mass calculation in SOFT = 2 contact..

Detailed Description

The Contact class allows you to create, modify, edit and contact cards. See the documentation below for more details.

Constructor

new Contact(Model*[Model]*, type*[string]*, id (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Contact object.

Arguments

Name	Type	Description
Model	Model	Model that Contact will be created in
type	string	Type of contact
id (optional)	integer	Contact number
heading (optional)	string	Title for the Contact

Return type

Contact object

Example

To create a new AUTOMATIC_GENERIC contact n model m with label 10 and title "Test contact"

```
var c = new Contact(m, "AUTOMATIC_GENERAL", 10, "Test contact");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a contact.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the contact

Return type

No return value

Example

To associate comment c to the contact c:

```
c.AssociateComment(c);
```

Blank()

Description

Blanks the contact

Arguments

No arguments

Return type

No return value

Example

To blank contact c:

```
c.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the contacts in model m:

```
Contact.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contacts will be blanked in
flag	Flag	Flag set on the contacts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the contacts in model m flagged with f:

```
Contact.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the contact is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if contact c is blanked:

```
if (c.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse contact c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the contact.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the contact

Return type

No return value

Example

To clear flag f for contact c:

```
c.ClearFlag(f);
```

Constrained(connection (optional)*[boolean]*)

Description

see if tied/spotweld contact uses constrained formulation

Arguments

Name	Type	Description
connection (optional)	boolean	if true will only consider contacts used for primer connections. The default is false.

Return type

logical

Example

To see if contact is of type tied and constrained

```
c.Constrained();
```

Copy(range (optional)*[boolean]*)

Description

Copies the contact.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Contact object

Example

To copy contact c into contact z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a contact.

Arguments

Name	Type	Description
Model	Model	Model that the contact will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Contact object (or null if not made)

Example

To start creating a contact in model m:

```
var c = Contact.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a contact.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the contact

Return type

No return value

Example

To detach comment c from the contact c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit contact c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for contact. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for contact c:

```
c.Error("My custom error");
```

FindInteractions() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use Contact.Interactions() instead.

Arguments

No arguments

Return type

No return value

First(Model*[Model]*) [static]

Description

Returns the first contact in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first contact in

Return type

Contact object (or null if there are no contacts in the model).

Example

To get the first contact in model m:

```
var c = Contact.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free contact label in the model. Also see Contact.LastFreeLabel(), Contact.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free contact label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Contact label.

Example

To get the first free contact label in model m:

```
var label = Contact.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the contacts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be flagged in
flag	Flag	Flag to set on the contacts

Return type

No return value

Example

To flag all of the contacts with flag f in model m:

```
Contact.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the contact is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the contact

Return type

true if flagged, false if not.

Example

To check if contact c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each contact in the model.
Note that ForEach has been designed to make looping over contacts as fast as possible and so has some limitations.
Firstly, a single temporary Contact object is created and on each function call it is updated with the current contact data. This means that you should not try to store the Contact object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new contacts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all contacts are in
func	function	Function to call for each contact
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the contacts in model m:

```
Contact.ForEach(m, test);
function test(c)
{
// c is Contact object
}
```

To call function test for all of the contacts in model m with optional object:

```
var data = { x:0, y:0 };
Contact.ForEach(m, test, data);
function test(c, extra)
{
// c is Contact object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Contact objects for all of the contacts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get contacts from

Return type

Array of Contact objects

Example

To make an array of Contact objects for all of the contacts in model m

```
var c = Contact.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a contact.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the contact c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Contact objects for all of the flagged contacts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get contacts from
flag	Flag	Flag set on the contacts that you want to retrieve

Return type

Array of Contact objects

Example

To make an array of Contact objects for all of the contacts in model m flagged with f

```
var c = Contact.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Contact object for a contact ID.

Arguments

Name	Type	Description
Model	Model	Model to find the contact in
number	integer	number of the contact you want the Contact object for

Return type

Contact object (or null if contact does not exist).

Example

To get the Contact object for contact 100 in model m

```
var c = Contact.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Contact property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Contact.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	contact property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Contact property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Contact property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Interactions(type (optional)*[constant]*)

Description

Returns an array of objects describing the interactions which can either be penetrations (slave nodes that are tied to or penetrate elements in the contact) or crossed edges (contact segments that cross).

Arguments

Name	Type	Description
type (optional)	constant	What type of interactions to return. Can be bitwise code of Contact.PENETRATIONS to return penetrations and Contact.CROSSED_EDGES to return crossed edges. If omitted penetrations will be returned.

Return type

Array of objects with the following properties:

Name	Type	Description
end	Array of reals	End coordinate of intersection line (for Contact.CROSSED_EDGES)
n1	Node object	Node 1 of master segment (for Contact.PENETRATIONS)

n2	Node object	Node 2 of master segment (for Contact.PENETRATIONS)
n3	Node object	Node 3 of master segment (for Contact.PENETRATIONS)
n4	Node object	Node 4 of master segment (for Contact.PENETRATIONS)
node	Node object	Penetrating node (for Contact.PENETRATIONS)
pen	real	Depth of penetration (for Contact.PENETRATIONS)
qthick	real	Remaining thickness ratio (for Contact.PENETRATIONS)
rthick	real	Remaining unpenetrated thickness (for Contact.PENETRATIONS)
s	real	s parametric coordinate of the slave node projected onto the shell (for Contact.PENETRATIONS)
shell	Shell object	Penetrated shell (for Contact.PENETRATIONS)
shell1	Shell object	First segment if shell (for Contact.CROSSED_EDGES)
shell2	Shell object	Second segment if shell (for Contact.CROSSED_EDGES)
solid	Solid object	Penetrated solid (for Contact.PENETRATIONS)
solid1	Solid object	First segment if solid (for Contact.CROSSED_EDGES)
solid2	Solid object	Second segment if solid (for Contact.CROSSED_EDGES)
start	Array of reals	Start coordinate of intersection line (for Contact.CROSSED_EDGES)
t	real	t parametric coordinate of the slave node projected onto the shell (for Contact.PENETRATIONS)
thick	real	Thickness of contact segment, i.e. 0.5*(t1+t2) (for Contact.PENETRATIONS)
thickshell	Tshell object	Penetrated thick shell (for Contact.PENETRATIONS)
thickshell1	Tshell object	First segment if thick shell (for Contact.CROSSED_EDGES)
thickshell2	Tshell object	Second segment if thick shell (for Contact.CROSSED_EDGES)
type	integer	The interaction type. Either Contact.PENETRATIONS or Contact.CROSSED_EDGES.

Example

To get the penetration interactions for contact c:

```
var interactions = c.Interactions();
for(i=0; i<interactions.length; i++)
{
    var type  = interactions[i].type; // Will be Contact.PENETRATIONS
    var node  = interactions[i].node;
    var shell = interactions[i].shell;
    var n1    = interactions[i].n1;
    var n2    = interactions[i].n2;
    var n3    = interactions[i].n3;
    var n4    = interactions[i].n4;
    var s     = interactions[i].s;
    var t     = interactions[i].t;
    var pen   = interactions[i].pen;
    var thick = interactions[i].thick;
    var rthick = interactions[i].rthick;
    var qthick = interactions[i].qthick;
    if(shell != undefined)
            ... process shell ...
}
```

To get the penetration and crossed edge interactions for contact c:

```
var interactions = c.Interactions(Contact.PENETRATIONS|Contact.CROSSED_EDGES);
for(i=0; i<interactions.length; i++)
{
    if (interactions[i].type == Contact.PENETRATIONS)
    {
        var node  = interactions[i].node;
        ...
    }
    else if (interactions[i].type == Contact.CROSSED_EDGES)
    {
        var start = interactions[i].start;
        ...
    }
}
```

Keyword()

Description

Returns the keyword for this Contact (*BOUNDARY_PRESCRIBED_MOTION_xxxx). **Note that a carriage return is not added**. See also Contact.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Contact c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Contact. **Note that a carriage return is not added**. See also Contact.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Contact c:

```
var cards = c.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last contact in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last contact in

Return type

Contact object (or null if there are no contacts in the model).

Example

To get the last contact in model m:

```
var c = Contact.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free contact label in the model. Also see Contact.FirstFreeLabel(), Contact.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free contact label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Contact label.

Example

To get the last free contact label in model m:

```
var label = Contact.LastFreeLabel(m);
```

Next()

Description

Returns the next contact in the model.

Arguments

No arguments

Return type

Contact object (or null if there are no more contacts in the model).

Example

To get the contact in model m after contact c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) contact label in the model. Also see Contact.FirstFreeLabel(), Contact.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free contact label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Contact label.

Example

To get the next free contact label in model m:

```
var label = Contact.NextFreeLabel(m);
```

PenCheck(flag*[Flag]*, eflag*[integer]*)

Description

Flags nodes that penetrate (or tie) in contact

Arguments

Name	Type	Description
flag	Flag	Flag to be set on penetrating (or tied) node.
eflag	integer	Optional flag for elements. If supplied, node will be flagged only if it penetrates (or ties to) an element that is flagged. Node and element flag may be the same.

Return type

zero if contact successfully checked

Example

To set flag f on slave nodes of Contact c which tie to elements flagged with f:

```
c.PenCheck(f, f);
```

PenCheckEdit(modal (optional)*[boolean]*, check_mode (optional)*[constant]*, mpp_threshold (optional)*[real]*, report_crossed_3d_elems (optional)*[boolean]*, contact_penchk_dup_shells (optional)*[constant]*)

Description

launches the interactive edit panel for penetration check on the con

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.
check_mode (optional)	constant	Check mode. Can be Model.MPP_MODE or Model.SMP_MODE. Default is set to the oa pref contact_check_mode
mpp_threshold (optional)	real	Can set the MPP threshold, by default this is set to the oa pref contact_mpp_ penetration_threshold
report_crossed_3d_ elems (optional)	boolean	Can set the value of reporting crossed elements to TRUE or FALSE, by default this is set to the oa pref report_crossed_3d_elems
contact_penchk_dup_ shells (optional)	constant	Duplicate shell treatment Can be Model.SHELL_AUTO, Model.SHELL_THICK or Model.SHELL_THIN. Default is set to the oa pref contact_penchk_dup_shells

Return type

No return value

Example

To launch an edit panel with modal set to TRUE, check_method set to MPP, mpp_threshold set to 1.123, report_ crossed_3d_elems set to true and contact_penchk_dup_shells set to thinnest always:

```
c.PenCheckEdit(true,Contact.MPP_METHOD,1.123,true,Contact.SHELL_THIN);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a contact.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only contacts from that model can be picked. If the argument is a Flag then only contacts that are flagged with *limit* can be selected. If omitted, or null, any contacts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.

button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Contact object (or null if not picked)

Example

To pick a contact from model m giving the prompt 'Pick contact from screen':

```
var c = Contact.Pick('Pick contact from screen', m);
```

Previous()

Description

Returns the previous contact in the model.

Arguments

No arguments

Return type

Contact object (or null if there are no more contacts in the model).

Example

To get the contact in model m before contact c:

```
var c = c.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the contacts in model m, from 1000000:

```
Contact.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contacts will be renumbered in
flag	Flag	Flag set on the contacts that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the contacts in model m flagged with f, from 1000000:

```
Contact.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select contacts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting contacts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only contacts from that model can be selected. If the argument is a Flag then only contacts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any contacts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of contacts selected or null if menu cancelled

Example

To select contacts from model m, flagging those selected with flag f, giving the prompt 'Select contacts':

```
Contact.Select(f, 'Select contacts', m);
```

To select contacts, flagging those selected with flag f but limiting selection to contacts flagged with flag l, giving the prompt 'Select contacts':

```
Contact.Select(f, 'Select contacts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the contact.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the contact

Return type

No return value

Example

To set flag f for contact c:

```
c.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the contact. The contact will be sketched until you either call Contact.Unsketch(), Contact.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the contact is sketched. If omitted redraw is true. If you want to sketch several contacts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch contact c:

```
c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged contacts in the model. The contacts will be sketched until you either call Contact.Unsketch(), Contact.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contacts will be sketched in
flag	Flag	Flag set on the contacts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the contacts are sketched. If omitted redraw is true. If you want to sketch flagged contacts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all contacts flagged with flag in model m:

```
Contact.SketchFlagged(m, flag);
```

StatusCheck()

Description

Checks sliding contact for crossed edges and penetrations

Arguments

No arguments

Return type

An array containing count of crossed edges, count of penetrations (note if a node penetrates more than one segment, it is only reported once here)

Example

To check Contact c:

```
var status = c.StatusCheck(); ncrossed = status[0]; npens = status[1]
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing contacts should be counted. If false or omitted referenced but undefined contacts will also be included in the total.

Return type

number of contacts

Example

To get the total number of contacts in model m:

```
var total = Contact.Total(m);
```

Unblank()

Description

Unblanks the contact

Arguments

No arguments

Return type

No return value

Example

To unblank contact c:

```
c.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the contacts in model m:

```
Contact.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged contacts will be unblanked in
flag	Flag	Flag set on the contacts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the contacts in model m flagged with f:

```
Contact.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all contacts will be unset in

flag	Flag	Flag to unset on the contacts

Return type

No return value

Example

To unset the flag f on all the contacts in model m:

```
Contact.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the contact.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the contact is unsketched. If omitted redraw is true. If you want to unsketch several contacts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch contact c:

```
c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all contacts.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the contacts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all contacts in model m:

```
Contact.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged contacts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contacts will be unsketched in
flag	Flag	Flag set on the contacts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the contacts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all contacts flagged with flag in model m:

```
Contact.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Contact object.

Example

To check if Contact property c.example is a parameter by using the Contact.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for contact. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for contact c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this contact.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for contact c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the Contact data in keyword format. Note that this contains the keyword header and the keyword cards. See also Contact.Keyword() and Contact.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Contact c in keyword format

```
var data = c.toString();
```

ContactGuidedCable class

The ContactGuidedCable class gives you access to define *CONTACT_GUIDED_CABLE cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ContactGuidedCable constants

Name	Description
ContactGuidedCable.PART	CONTACT is *CONTACT_GUIDED_CABLE.
ContactGuidedCable.SET_PART	CONTACT is *CONTACT_GUIDED_CABLE_SET.

ContactGuidedCable properties

Name	Type	Description
cid	integer	ContactGuidedCable number.
endtol	real	Tolerance, in length units.
exists	logical	true if ContactGuidedCable exists, false if referred to but not defined. (read only)
fric	real	Contact friction.
heading	string	ContactGuidedCable heading
id	logical	TRUE if _ID option is set, FALSE if not
include	integer	The Include file number that the ContactGuidedCable is in.
model	integer	The Model number that the contact guided_cable is in.
nsid	integer	Node Set ID that guides the 1D elements.
pid	integer	Part ID or Part Set ID
ptype	constant	The Contact Part type. Can be ContactGuidedCable.PART or ContactGuidedCable.SET_PART.
soft	integer	Flag for soft constraint option. Set to 1 for soft constraint.
ssfac	real	Stiffness scale factor for penalty stiffness value. The default value is unity. This applies to SOFT set to 0 and 1.

Detailed Description

The ContactGuidedCable class allows you to create, modify, edit and manipulate *CONTACT_GUIDED_CABLE cards. See the documentation below for more details.

Constructor

new ContactGuidedCable(Model*[Model]*, ptype*[constant]*, nsid*[integer]*, pid*[integer]*, soft (optional)*[integer]*, ssfac (optional)*[real]*, fric (optional)*[real]*, cid (optional)*[integer]*, heading (optional)*[string]*, endtol (optional)*[real]*)

Description

Create a new ContactGuidedCable object.

Arguments

Name	Type	Description
Model	Model	Model that ContactGuidedCable will be created in
ptype	constant	Specify the type of ContactGuidedCable (Can be ContactGuidedCable.PART or ContactGuidedCable.SET_PART
nsid	integer	Node Set ID that guides the 1D elements.
pid	integer	Part ID or Part Set ID
soft (optional)	integer	Flag for soft constraint option. Set to 1 for soft constraint.

ssfac (optional)	real	Stiffness scale factor for penalty stiffness value. The default value is unity. This applies to SOFT set to 0 and 1.
fric (optional)	real	Contact friction.
cid (optional)	integer	ContactGuidedCable number (Same as label).
heading (optional)	string	ContactGuidedCable heading (Same as title).
endtol (optional)	real	Tolerance, in length units.

Return type

ContactGuidedCable object

Example

To create a new contact guided_cable in model m, of ptype PART, with nsid 100, pid 10, soft 1 and ssfac 4.5.

```
var c_g_c = new ContactGuidedCable(m, ContactGuidedCable.PART, 100, 10, 1, 4.5);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a contact guided_cable.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the contact guided_cable

Return type

No return value

Example

To associate comment c to the contact guided_cable c_g_c:

```
c_g_c.AssociateComment(c);
```

Blank()

Description

Blanks the contact guided_cable

Arguments

No arguments

Return type

No return value

Example

To blank contact guided_cable c_g_c:

```
c_g_c.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the contact guided_cables in model m:

```
ContactGuidedCable.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contact guided_cables will be blanked in
flag	Flag	Flag set on the contact guided_cables that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the contact guided_cables in model m flagged with f:

```
ContactGuidedCable.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the contact guided_cable is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if contact guided_cable c_g_c is blanked:

```
if (c_g_c.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the contact guided_cable.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the contact guided_cable

Return type

No return value

Example

To clear flag f for contact guided_cable c_g_c:

```
c_g_c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the contact guided_cable.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ContactGuidedCable object

Example

To copy contact guided_cable c_g_c into contact guided_cable z:

```
var z = c_g_c.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a contact guided_cable.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the contact guided_cable

Return type

No return value

Example

To detach comment c from the contact guided_cable c_g_c:

```
c_g_c.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for contact guided_cable. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for contact guided_cable c_g_c:

```
c_g_c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first contact guided_cable in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first contact guided_cable in

Return type

ContactGuidedCable object (or null if there are no contact guided_cables in the model).

Example

To get the first contact guided_cable in model m:

```
var c_g_c = ContactGuidedCable.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free contact guided_cable label in the model. Also see ContactGuidedCable.LastFreeLabel(), ContactGuidedCable.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free contact guided_cable label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

ContactGuidedCable label.

Example

To get the first free contact guided_cable label in model m:

```
var label = ContactGuidedCable.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the contact guided_cables in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be flagged in
flag	Flag	Flag to set on the contact guided_cables

Return type

No return value

Example

To flag all of the contact guided_cables with flag f in model m:

```
ContactGuidedCable.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the contact guided_cable is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the contact guided_cable

Return type

true if flagged, false if not.

Example

To check if contact guided_cable c_g_c has flag f set on it:

```
if (c_g_c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each contact guided_cable in the model.
Note that ForEach has been designed to make looping over contact guided_cables as fast as possible and so has some limitations.
Firstly, a single temporary ContactGuidedCable object is created and on each function call it is updated with the current contact guided_cable data. This means that you should not try to store the ContactGuidedCable object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new contact guided_cables inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables are in
func	function	Function to call for each contact guided_cable
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the contact guided_cables in model m:

```
ContactGuidedCable.ForEach(m, test);
function test(c_g_c)
{
// c_g_c is ContactGuidedCable object
}
```

To call function test for all of the contact guided_cables in model m with optional object:

```
var data = { x:0, y:0 };
ContactGuidedCable.ForEach(m, test, data);
function test(c_g_c, extra)
{
// c_g_c is ContactGuidedCable object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of ContactGuidedCable objects for all of the contact guided_cables in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get contact guided_cables from

Return type

Array of ContactGuidedCable objects

Example

To make an array of ContactGuidedCable objects for all of the contact guided_cables in model m

```
var c_g_c = ContactGuidedCable.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a contact guided_cable.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the contact guided_cable c_g_c:

```
var comm_array = c_g_c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of ContactGuidedCable objects for all of the flagged contact guided_cables in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get contact guided_cables from
flag	Flag	Flag set on the contact guided_cables that you want to retrieve

Return type

Array of ContactGuidedCable objects

Example

To make an array of ContactGuidedCable objects for all of the contact guided_cables in model m flagged with f

```
var c_g_c = ContactGuidedCable.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ContactGuidedCable object for a contact guided_cable ID.

Arguments

Name	Type	Description
Model	Model	Model to find the contact guided_cable in
number	integer	number of the contact guided_cable you want the ContactGuidedCable object for

Return type

ContactGuidedCable object (or null if contact guided_cable does not exist).

Example

To get the ContactGuidedCable object for contact guided_cable 100 in model m

```
var c_g_c = ContactGuidedCable.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a ContactGuidedCable property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ContactGuidedCable.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	contact guided_cable property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ContactGuidedCable property c_g_c.example is a parameter:

```
Options.property_parameter_names = true;
if (c_g_c.GetParameter(c_g_c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ContactGuidedCable property c_g_c.example is a parameter by using the GetParameter method:

```
if (c_g_c.ViewParameters().GetParameter(c_g_c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this ContactGuidedCable (*contact_guided_cable). **Note that a carriage return is not added**. See also ContactGuidedCable.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for ContactGuidedCable c_g_c:

```
var key = c_g_c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the ContactGuidedCable. **Note that a carriage return is not added**. See also ContactGuidedCable.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for ContactGuidedCable c_g_c:

```
var cards = c_g_c.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last contact guided_cable in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last contact guided_cable in

Return type

ContactGuidedCable object (or null if there are no contact guided_cables in the model).

Example

To get the last contact guided_cable in model m:

```
var c_g_c = ContactGuidedCable.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free contact guided_cable label in the model. Also see ContactGuidedCable.FirstFreeLabel(), ContactGuidedCable.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free contact guided_cable label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

ContactGuidedCable label.

Example

To get the last free contact guided_cable label in model m:

```
var label = ContactGuidedCable.LastFreeLabel(m);
```

Next()

Description

Returns the next contact guided_cable in the model.

Arguments

No arguments

Return type

ContactGuidedCable object (or null if there are no more contact guided_cables in the model).

Example

To get the contact guided_cable in model m after contact guided_cable c_g_c:

```
var c_g_c = c_g_c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) contact guided_cable label in the model. Also see ContactGuidedCable.FirstFreeLabel(), ContactGuidedCable.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free contact guided_cable label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

ContactGuidedCable label.

Example

To get the next free contact guided_cable label in model m:

```
var label = ContactGuidedCable.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a contact guided_cable.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only contact guided_cables from that model can be picked. If the argument is a Flag then only contact guided_cables that are flagged with *limit* can be selected. If omitted, or null, any contact guided_cables from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

ContactGuidedCable object (or null if not picked)

Example

To pick a contact guided_cable from model m giving the prompt 'Pick contact guided_cable from screen':

```
var c_g_c = ContactGuidedCable.Pick('Pick contact guided_cable from screen', m);
```

Previous()

Description

Returns the previous contact guided_cable in the model.

Arguments

No arguments

Return type

ContactGuidedCable object (or null if there are no more contact guided_cables in the model).

Example

To get the contact guided_cable in model m before contact guided_cable c_g_c:

```
var c_g_c = c_g_c.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the contact guided_cables in model m, from 1000000:

```
ContactGuidedCable.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contact guided_cables will be renumbered in

flag	Flag	Flag set on the contact guided_cables that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the contact guided_cables in model m flagged with f, from 1000000:

```
ContactGuidedCable.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select contact guided_cables using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting contact guided_cables
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only contact guided_cables from that model can be selected. If the argument is a Flag then only contact guided_cables that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any contact guided_cables can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of contact guided_cables selected or null if menu cancelled

Example

To select contact guided_cables from model m, flagging those selected with flag f, giving the prompt 'Select contact guided_cables':

```
ContactGuidedCable.Select(f, 'Select contact guided_cables', m);
```

To select contact guided_cables, flagging those selected with flag f but limiting selection to contact guided_cables flagged with flag l, giving the prompt 'Select contact guided_cables':

```
ContactGuidedCable.Select(f, 'Select contact guided_cables', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the contact guided_cable.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the contact guided_cable

Return type

No return value

Example

To set flag f for contact guided_cable c_g_c:

```
c_g_c.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the contact guided_cable. The contact guided_cable will be sketched until you either call ContactGuidedCable.Unsketch(), ContactGuidedCable.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the contact guided_cable is sketched. If omitted redraw is true. If you want to sketch several contact guided_cables and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch contact guided_cable c_g_c:

```
c_g_c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged contact guided_cables in the model. The contact guided_cables will be sketched until you either call ContactGuidedCable.Unsketch(), ContactGuidedCable.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged contact guided_cables will be sketched in
flag	Flag	Flag set on the contact guided_cables that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the contact guided_cables are sketched. If omitted redraw is true. If you want to sketch flagged contact guided_cables several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all contact guided_cables flagged with flag in model m:

```
ContactGuidedCable.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing contact guided_cables should be counted. If false or omitted referenced but undefined contact guided_cables will also be included in the total.

Return type

number of contact guided_cables

Example

To get the total number of contact guided_cables in model m:

```
var total = ContactGuidedCable.Total(m);
```

Unblank()

Description

Unblanks the contact guided_cable

Arguments

No arguments

Return type

No return value

Example

To unblank contact guided_cable c_g_c:

```
c_g_c.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the contact guided_cables in model m:

```
ContactGuidedCable.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged contact guided_cables will be unblanked in
flag	Flag	Flag set on the contact guided_cables that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the contact guided_cables in model m flagged with f:

```
ContactGuidedCable.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all contact guided_cables will be unset in
flag	Flag	Flag to unset on the contact guided_cables

Return type

No return value

Example

To unset the flag f on all the contact guided_cables in model m:

```
ContactGuidedCable.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the contact guided_cable.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the contact guided_cable is unsketched. If omitted redraw is true. If you want to unsketch several contact guided_cables and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch contact guided_cable c_g_c:

```
c_g_c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all contact guided_cables.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the contact guided_cables are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all contact guided_cables in model m:

```
ContactGuidedCable.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged contact guided_cables in the model.

Arguments

Name	Type	Description
Model	Model	Model that all contact guided_cables will be unsketched in

flag	Flag	Flag set on the contact guided_cables that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the contact guided_cables are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all contact guided_cables flagged with flag in model m:

```
ContactGuidedCable.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ContactGuidedCable object.

Example

To check if ContactGuidedCable property c_g_c.example is a parameter by using the ContactGuidedCable.GetParameter() method:

```
if (c_g_c.ViewParameters().GetParameter(c_g_c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for contact guided_cable. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for contact guided_cable c_g_c:

```
c_g_c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this contact guided_cable.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for contact guided_cable c_g_c:

```
var xrefs = c_g_c.Xrefs();
```

---

toString()

Description

Creates a string containing the ContactGuidedCable data in keyword format. Note that this contains the keyword header and the keyword cards. See also ContactGuidedCable.Keyword() and ContactGuidedCable.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for ContactGuidedCable c_g_c in keyword format

```
var s = c_g_c.toString();
```

Control class

The Control class gives you access to control cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Control properties

Name	Type	Description
accuracy	Object	*CONTROL_ACCURACY card
acoustic	Object	*CONTROL_ACOUSTIC card
adapstep	Object	*CONTROL_ADAPSTEP card
adaptive	Object	*CONTROL_ADAPTIVE card
adaptive_curve	Object	*CONTROL_ADAPTIVE_CURVE card
airbag	Object	*CONTROL_AIRBAG card
ale	Object	*CONTROL_ALE card
bulk_viscosity	Object	*CONTROL_BULK_VISCOSITY card
check	Object	*CONTROL_CHECK card
coarsen	Object	*CONTROL_COARSEN card
constrained	Object	*CONTROL_CONSTRAINED card
contact	Object	*CONTROL_CONTACT card
coupling	Object	*CONTROL_COUPLING card
cpm	Object	*CONTROL_CPM card
cpu	Object	*CONTROL_CPU card
debug	Object	*CONTROL_DEBUG card
discrete_element	Object	*CONTROL_DISCRETE_ELEMENT card
dynamic_relaxation	Object	*CONTROL_DYNAMIC_RELAXATION card
efg	Object	*CONTROL_EFG card
energy	Object	*CONTROL_ENERGY card
explicit_thermal	Object	*CONTROL_EXPLICIT_THERMAL_PROPERTIES card
explicit_thermal_ale_coupling	Object	*CONTROL_EXPLICIT_THERMAL_ALE_COUPLING card
explicit_thermal_boundary	Object	*CONTROL_EXPLICIT_THERMAL_BOUNDARY card
explicit_thermal_contact	Object	*CONTROL_EXPLICIT_THERMAL_CONTACT card
explicit_thermal_initial	Object	*CONTROL_EXPLICIT_THERMAL_INITIAL card

explicit_thermal_output	Object	*CONTROL_EXPLICIT_THERMAL_OUTPUT card
explicit_thermal_solver	Object	*CONTROL_EXPLICIT_THERMAL_SOLVER card
explosive_shadow	Object	*CONTROL_EXPLOSIVE_SHADOW card
forming_bestfit	Object	*CONTROL_FORMING_CONTROL_FORMING_BESTFIT card
forming_initial_thickness	Object	*CONTROL_FORMING_INITIAL_THICKNESS card
forming_maxid	Object	*CONTROL_FORMING_MAXID card
forming_position	Object	*CONTROL_FORMING_POSITION card
forming_pre_bending	Object	*CONTROL_FORMING_PRE_BENDING card
forming_projection	Object	*CONTROL_FORMING_PROJECTION card
forming_remove_adaptive_constraints	Object	*CONTROL_FORMING_REMOVE_ADAPTIVE_CONSTRAINTS card
forming_shell_to_tshell	Object	*CONTROL_FORMING_SHELL_TO_TSHELL card
forming_stoning	Object	*CONTROL_FORMING_STONING card
forming_strain_ratio_smooth	Object	*CONTROL_FORMING_STRAIN_RATIO_SMOOTH card
forming_template	Object	*CONTROL_FORMING_TEMPLATE card
forming_toleranc	Object	*CONTROL_FORMING_TOLERANC card
forming_travel	Object	*CONTROL_FORMING_TRAVEL card
forming_trim_merge	Object	*CONTROL_FORMING_TRIM_MERGE card
forming_trim_solid_refinement	Object	*CONTROL_FORMING_TRIM_SOLID_REFINEMENT card
forming_unflanging	Object	*CONTROL_FORMING_UNFLANGING card
forming_user	Object	*CONTROL_FORMING_USER card
frequency_domain	Object	*CONTROL_FREQUENCY_DOMAIN card
frequency_response_function	Object	*CONTROL_FREQUENCY_RESPONSE_FUNCTION card
hourglass	Object	*CONTROL_HOURGLASS card
implicit_auto	Object	*CONTROL_IMPLICIT_AUTO card
implicit_buckle	Object	*CONTROL_IMPLICIT_BUCKLE card
implicit_consistent_mass	Object	*CONTROL_IMPLICIT_CONSISTENT_MASS card
implicit_dynamics	Object	*CONTROL_IMPLICIT_DYNAMICS card
implicit_eigenvalue	Object	*CONTROL_IMPLICIT_EIGENVALUE card
implicit_explicit_hybrid	Object	*CONTROL_IMPLICIT_EXPLICT_HYBRID card
implicit_forming	Object	*CONTROL_IMPLICIT_FORMING card
implicit_general	Object	*CONTROL_IMPLICIT_GENERAL card
implicit_inertia_relief	Object	*CONTROL_IMPLICIT_INERTIA_RELIEF card
implicit_joints	Object	*CONTROL_IMPLICIT_JOINTS card
implicit_modal_dynamic	Object	*CONTROL_IMPLICIT_MODAL_DYNAMIC card

implicit_modal_dynamic_damping	Object	*CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING card
implicit_modes	Object	*CONTROL_IMPLICIT_MODES card
implicit_ordering	Object	*CONTROL_IMPLICIT_ORDERING card
implicit_residual_vector	Object	*CONTROL_IMPLICIT_RESIDUAL_VECTOR card
implicit_solution	Object	*CONTROL_IMPLICIT_SOLUTION card
implicit_solver	Object	*CONTROL_IMPLICIT_SOLVER card
implicit_stabilization	Object	*CONTROL_IMPLICIT_STABILIZATION card
implicit_static_condensation	Object	*CONTROL_IMPLICIT_STATIC_CONDENSATION card
implicit_termination	Object	*CONTROL_IMPLICIT_TERMINATION card
mpp_contact_groupable	Object	*CONTROL_MPP_CONTACT_GROUPABLE card
mpp_decomposition_automatic	Object	*CONTROL_MPP_DECOMPOSITION_AUTOMATIC card
mpp_decomposition_bagref	Object	*CONTROL_MPP_DECOMPOSITION_BAGREF card
mpp_decomposition_check_speed	Object	*CONTROL_MPP_DECOMPOSITION_CHECK_SPEED card
mpp_decomposition_contact_isolate	Object	*CONTROL_MPP_DECOMPOSITION_CONTACT_ISOLATE card
mpp_decomposition_disable_unref_curves	Object	*CONTROL_MPP_DECOMPOSITION_DISABLE_UNREF_CURVES card
mpp_decomposition_distribute_ale_elements	Object	*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_ALE_ELEMENTS card
mpp_decomposition_distribute_sph_elements	Object	*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_SPH_ELEMENTS card
mpp_decomposition_elcost	Object	*CONTROL_MPP_DECOMPOSITION_ELCOST card
mpp_decomposition_file	Object	*CONTROL_MPP_DECOMPOSITION_FILE card
mpp_decomposition_flag_stress_strain_curve	Object	*CONTROL_MPP_DECOMPOSITION_FLAG_STRESS_STRAIN_CURVE card
mpp_decomposition_method	Object	*CONTROL_MPP_DECOMPOSITION_METHOD card
mpp_decomposition_numproc	Object	*CONTROL_MPP_DECOMPOSITION_NUMPROC card
mpp_decomposition_outdecomp	Object	*CONTROL_MPP_DECOMPOSITION_OUTDECOMP card
mpp_decomposition_rcblog	Object	*CONTROL_MPP_DECOMPOSITION_RCBLOG card
mpp_decomposition_scale_contact_cost	Object	*CONTROL_MPP_DECOMPOSITION_SCALE_CONTACT_COST card
mpp_decomposition_scale_factor_sph	Object	*CONTROL_MPP_DECOMPOSITION_SCALE_FACTOR_SPH card
mpp_decomposition_show	Object	*CONTROL_MPP_DECOMPOSITION_SHOW card
mpp_decomposition_transformation	Object	*CONTROL_MPP_DECOMPOSITION_TRANSFORMATION card
mpp_io_binoutonly	Object	*CONTROL_MPP_IO_BINOUTONLY card
mpp_io_lstc_reduce	Object	*CONTROL_MPP_IO_LSTC_REDUCE card
mpp_io_nod3dump	Object	*CONTROL_MPP_IO_NOD3DUMP card

mpp_io_nodump	Object	*CONTROL_MPP_IO_NODUMP card
mpp_io_nofail	Object	*CONTROL_MPP_IO_NOFAIL card
mpp_io_nofull	Object	*CONTROL_MPP_IO_NOFULL card
mpp_io_swapbytes	Object	*CONTROL_MPP_IO_SWAPBYTES card
mpp_mat_model_driver	Object	*CONTROL_MPP_MATERIAL_MODEL_DRIVER card
mpp_rebalance	Object	*CONTROL_MPP_REBALANCE card
nonlocal	Object	*CONTROL_NONLOCAL card
output	Object	*CONTROL_OUTPUT card
parallel	Object	*CONTROL_PARALLEL card
pore_air	Object	*CONTROL_PORE_AIR card
pore_fluid	Object	*CONTROL_PORE_FLUID card
pwp_auto_tmf	Object	*CONTROL_PWP_AUTO_TMF card
pzelectric	Object	*CONTROL_PZELECTRIC card
ref_config	Object	*CONTROL_REFERENCE_CONFIGURATION card
remesh	Object	*CONTROL_REMESHING card
rigid	Object	*CONTROL_RIGID card
shell	Object	*CONTROL_SHELL card
solid	Object	*CONTROL_SOLID card
solution	Object	*CONTROL_SOLUTION card
sph	Object	*CONTROL_SPH card
sph_incompressible	Object	*CONTROL_SPH_INCOMPRESSIBLE card
spotweld_beam	Object	*CONTROL_SPOTWELD_BEAM card
staged_construction	Object	*CONTROL_STAGED_CONSTRUCTION card
start	Object	*CONTROL_START card
steady_state_rolling	Object	*CONTROL_STEADY_STATE_ROLLING card
structured	Object	*CONTROL_STRUCTURED card
termination	Object	*CONTROL_TERMINATION card
thermal_eigenvalue	Object	*CONTROL_THERMAL_EIGENVALUE card
thermal_forming	Object	*CONTROL_THERMAL_FORMING card
thermal_nonlinear	Object	*CONTROL_THERMAL_NONLINEAR card
thermal_solver	Object	*CONTROL_THERMAL_SOLVER card
thermal_timestep	Object	*CONTROL_THERMAL_TIMESTEP card
timestep	Object	*CONTROL_TIMESTEP card
units	Object	*CONTROL_UNITS card
vibro_acoustic	Object	*CONTROL_VIBRO_ACOUSTIC card

Detailed Description

The Control class allows you to create, modify, edit and manipulate control cards. Unlike other classes there is no constructor and there are no functions. Instead a Control object is available as the control property of a Model object. This object allows you to access all of the control cards.
For example, to activate control card *CONTROL_TERMINATION in model m and set endtim to 0.1.

```
m.control.termination.exists = true;
m.control.termination.endtim = 0.1;
```

See the properties for more details.

*CONTROL_ACCURACY

Properties for *CONTROL_ACCURACY

Name	Type	Description
exacc	real	Explicit accuracy parameter
exists	logical	true if control card exists
iacc	integer	Implicit accuracy flag
include	integer	The Include file number that the control card is in.
inn	integer	Invariant node numbering for shell element
osu	integer	Objective stress update for large timestep
pidosu	integer	Part set id for objective stress updates

*CONTROL_ACOUSTIC

Properties for *CONTROL_ACOUSTIC

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
macdvp	logical	Acoustic nodal motions will be calculated or not.

*CONTROL_ADAPSTEP

Properties for *CONTROL_ADAPSTEP

Name	Type	Description
dfactr	real	Incremental increase in factin
exists	logical	true if control card exists
factin	real	Initial relaxation factor for contact force
include	integer	The Include file number that the control card is in.

*CONTROL_ADAPTIVE

Properties for *CONTROL_ADAPTIVE

Name	Type	Description
adpass	integer	1 or 2 pass adaptivity flag
adpctl	real	Adaptivity error tolerance in degrees for activating fusion
adpene	real	Nodal penetration at which to refine elem
adperr	integer	Options for recovery techniques and error estimators
adpfreq	real	Time interval between refinements
adpopt	integer	Adaptive options
adpsize	real	Min element edge size for adaptivity
adpth	real	Absolute shell thickness below which remeshing should begin
adptol	real	Adaptive error tolerance (degrees)
adptyp	integer	Adaptive options
cbirth	real	Birth time for adaptive fusion
cdeath	real	Death time for adaptive fusion
cnla	real	Limit angle for corner nodes
d3trace	integer	Flag for writing out d3plot state
exists	logical	true if control card exists
iadpcl	integer	Fission level that fusion will start at
iadpe90	integer	Maximum no. of elements covering 90degree of radii
iadpgh	integer	Fiffion flag for neighbour splitting
ifsand	integer	Flag for forming of sandwiched parts with adaptive blank mesh
include	integer	The Include file number that the control card is in.
ioflag	integer	Flag to generate adaptive mesh
ireflg	integer	Uniform refinement level. Loadcurve if negative
lcadp	integer	Loadcurve: Adaptive interval vs time
lclvl	integer	Loadcurve of maximum refinement level vs. time
maxel	integer	Max number of elements for adaptivity
maxlvl	integer	Max #refinement levels
memory	integer	Memory limit beyond which adaptivity will cease
mmm2d	integer	Option for merging common boundaries of all adapted materials
ncfreq	integer	Frequency of fission to fusion steps
orient	integer	Flag to set the global orientation of a forming contact
tbirth	real	Birth time for adaptivity
tdeath	real	Death time for adaptivity

*CONTROL_ADAPTIVE_CURVE

Properties for *CONTROL_ADAPTIVE_CURVE

Name	Type	Description
exists	logical	true if control card exists
idset	integer	Shell/Part set id
include	integer	The Include file number that the control card is in.
itriopt	integer	Refinement option for enclosed area of trim curve
itype	integer	Set type
n	integer	Refinement option
smin	real	Element dimension limit for refining

*CONTROL_AIRBAG

Properties for *CONTROL_AIRBAG

Name	Type	Description
ckerr	integer	Flag to check and report open edge of CV airbag
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_ALE

Properties for *CONTROL_ALE

Name	Type	Description
aafac	real	ALE advection factor
afac	real	Smoothing weight factor: simple average
beamin	real	Flag for aligning beam dynamics
bfac	real	Smoothing weight factor: volume weighting
bndflx	integer	Multi-Material ALE group set if positive or -1
cfac	real	Smoothing weight factor: isoparametric
checkr	real	Parameter for ALE pressure locking
dct	integer	Default continuum treatment
dfac	real	Smoothing weight factor: equipotential
dtmufac	real	Scale time step called DTMU
ebc	integer	Automatic Euler boundary condition
efac	real	Smoothing weight factor: equipotential
end	real	End time for smoothing
exists	logical	true if control card exists
ialedr	integer	Include ALE computations in the dynamic relaxation analysis
imascl	integer	Flag for mass scaling for ALE parts
include	integer	The Include file number that the control card is in.
meth	integer	Advection method

minmas	real	Factor of the minimum mass allowed in an element
mmgpref	integer	Selects the method that is used to include a reference pressure in a calculation involving ALE multi-material groups
nadv	integer	Number of cycles between advections
nbkt	integer	Number of Lagrangian cycles between bucket sort searches
ncpl	integer	Number of Lagrangian cycles between coupling calculations
nsidebc	integer	Optional excluded node set
optimpp	integer	Optimize the MPP communications (Range 0/1)
pdifmx	real	Max pressure difference for stress zeroing
pref	real	ref pressure on boundary
prit	integer	Pressure equalibrium flag
start	real	Start time for smoothing
vfact	real	Void factor

*CONTROL_BULK_VISCOSITY

Properties for *CONTROL_BULK_VISCOSITY

Name	Type	Description
btype	integer	beam bulk viscosity type
exists	logical	true if control card exists
ibq	integer	Default bulk viscosity type (m#PR035)
include	integer	The Include file number that the control card is in.
q1	real	Default linear viscosity coefficient
q2	real	Default quadratic viscosity coefficient
tstype	integer	Thick shell bulk viscosity type

*CONTROL_CHECK

Properties for *CONTROL_CHECK

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
solitary	integer	TRUE if a plain (no _SHELL suffix) card exists

*CONTROL_COARSEN

Properties for *CONTROL_COARSEN

Name	Type	Description
angle	real	Permitted angle between neighbours

exists	logical	true if control card exists
icoarse	integer	On/Off flag
include	integer	The Include file number that the control card is in.
n1	integer	Optional seed node ID 1
n2	integer	Optional seed node ID 2
n3	integer	Optional seed node ID 3
n4	integer	Optional seed node ID 4
n5	integer	Optional seed node ID 5
n6	integer	Optional seed node ID 6
n7	integer	Optional seed node ID 7
n8	integer	Optional seed node ID 8
nseed	integer	#extra "seed" nodes below
psid	integer	excluded part set
smax	real	Maximum element size

*CONTROL_CONSTRAINED

Properties for *CONTROL_CONSTRAINED

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
sprchk	integer	Flag to check and report open edge of CV constrained

*CONTROL_CONTACT

Properties for *CONTROL_CONTACT

Name	Type	Description
dfric	real	Default dynamic coefficient of friction
ecdt	integer	Timestep override for eroding contacts
edc	real	Default exponential decay coefficient
enmass	integer	Treatment of mass of eroded nodes
exists	logical	true if control card exists
frceng	integer	Flag to calculate internal friction energy
ftall	integer	output contact forces to rcforc
icov	integer	Invokes the covariant formulation of Konyukhov and Schweizerhof
igactc	integer	option to use isogeometric shells for contact detection
ignore	integer	Ignore initial penetrations flag
include	integer	The Include file number that the control card is in.

interm	integer	Intermittent searching flag for old contacts
islchk	integer	Initial penetration check flag
isym	integer	symmetry option. Node set if negative
ithcnt	integer	thermal contact heat transfer mode
ithoff	integer	Flag for offsetting thermal contact surfaces for thick thermal shells
nsbcs	integer	#cycles between 3D bucket sorts
nserod	integer	erosion option
orien	integer	Automatic contact segment orientation flag
outseg	integer	Spotweld output flag
pen_sf	real	Default local penalty scale factor
penopt	integer	Penalty stiffness option flag
pstiff	integer	method for penalty stiff calc
ptscl	real	scale factor on the contact stress exerted onto shells
rwgaps	integer	flag for gap stiffness
rwgdth	real	death time for gap stiffness
rwksf	real	penalty scale factor
rwpnal	real	Scale factor for rigid wall penalties
sfric	real	Default static coefficient of friction
shledg	integer	Flag for assuming edge shape for shells
shlthk	integer	Shell thickness consideration flag
shltrw	real	Shell thickness scale factor
skiprwg	integer	Display rigidwall flag
slsfac	real	Scale factor for sliding penalties
spotdel	integer	Spotweld deletion flag
spothin	real	Optional thickness scale factor
spotstp	integer	Error termination flag on unfound spotweld
ssthk	integer	Shell thickness use flag for type 4 contacts
swradf	real	Spot weld radius scale factor
tdcnof	integer	tied constraint offset contact update option
th	real	Default contact thickness
th_sf	real	Default thickness scale factor
thkchg	integer	Consider shell thickness change flag
tiedprj	integer	Projection bypass flag for TIED_ types
usrfrc	integer	Storage for user-controlled friction subroutine
usrstr	integer	Storage for user-controlled control subroutine
vfc	real	Default viscous friction coefficient
xpene	real	Surface max penetration check multiplier

*CONTROL_COUPLING

Properties for *CONTROL_COUPLING

Name	Type	Description
exists	logical	true if control card exists
flipx	integer	Flag to flip X coords
flipy	integer	Flag to flip Y coords
flipz	integer	Flag to flip Z coords
include	integer	The Include file number that the control card is in.
subcyl	integer	Subcycling flag
timidl	real	Idle time value
unforc	real	Force conversion factor
unleng	real	Length conversion factor
untime	real	Time conversion factor

*CONTROL_CPM

Properties for *CONTROL_CPM

Name	Type	Description
blkv	integer	Flag to allocate additional memory for contact nodal forces
cpmerr	integer	Disable/enable error checking
cpmmf	integer	Flag to consider airbag system velocity
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ncpmout	integer	Control CPM output database to d3plot
ncpmts	integer	Timestep size estimation
np2p	integer	Number of cycles for repartition particles
sffdc	real	Scale factor of force decay constant

*CONTROL_CPU

Properties for *CONTROL_CPU

Name	Type	Description
cputim	real	Max permitted cpu time
exists	logical	true if control card exists
iglst	integer	glstat data flag
include	integer	The Include file number that the control card is in.

*CONTROL_DEBUG

Properties for *CONTROL_DEBUG

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_DISCRETE_ELEMENT

Properties for *CONTROL_DISCRETE_ELEMENT

Name	Type	Description
ang	real	contact angle
bt	real	Birth time
cap	integer	dry/wet particle flag
cp	real	DES thermal property
dc	real	Exponential decay coefficient
dt	real	Death time
exists	logical	true if control card exists
fricd	real	Dynamic coefficient of friction
fricr	real	rolling friction coefficient
frics	real	friction coefficient
gamma	real	liquid surface tension
gap	real	parameter affecting spatial limit of liquid bridge
idesoft	integer	Flag for soft constraint formulation
ignore	integer	Ignore penetration flag
include	integer	The Include file number that the control card is in.
iskip	integer	Flag for skipping the calculation of contact force between DES:
lnorm	integer	LCID that defines the function for normal stiffness vs norm pen ratio
lshear	integer	LCID that defines the function for shear stiffness vs norm pen ratio
maxnei	integer	Number of neighbors to be tracked for DES contact and capillary force calculation
nbuf	integer	Asynchronous scheme and memory buffer option
ncrb	integer	Rebalancing frequency
ndamp	real	normal damping coefficient
normk	real	scale factor for normal spring constant
parallel	integer	Option to force calculation of bonded DES
sheark	real	ratio between sheark/normk
sofscl	real	Scale factor applied to the contact stiffness
tc	real	DES thermal property
tdamp	real	tangential damping coefficient
tfac	real	DES thermal property
vol	real	volume fraction
vtk	integer	max number of subcycling cycles

*CONTROL_DYNAMIC_RELAXATION

Properties for *CONTROL_DYNAMIC_RELAXATION

Name	Type	Description
drfctr	real	Dyn relaxation factor
drpset	integer	Part set used to check for convergence
drterm	real	Optional DR termination time
drtol	real	Convergence tolerance
edttl	real	Convergence tolerance on auto control
exists	logical	true if control card exists
idrflg	integer	Stress initialisation flag
include	integer	The Include file number that the control card is in.
irelal	integer	Automatic control flag
nrcyck	integer	#iterations between convergence checks
tssfdr	real	Optional timestep factor during DR

*CONTROL_EFG

Properties for *CONTROL_EFG

Name	Type	Description
etol	real	Error tolerance in the IMLM
exists	logical	true if control card exists
hsort	integer	Not used
ideb	integer	Output internal debug message
idila	integer	dilation param
imlm	integer	Choice for matrix operation, linear solving and memory usage
include	integer	The Include file number that the control card is in.
inint	integer	Factor needed for the estimation of maximum workspace used during initialization
ispline	integer	kernel function
ssort	integer	Flag for automatic sort of background triangular shells

*CONTROL_ENERGY

Properties for *CONTROL_ENERGY

Name	Type	Description
exists	logical	true if control card exists
hgen	integer	Hourglass energy calc flag
include	integer	The Include file number that the control card is in.

irgen	integer	Initial reference geometry calc flag
rwen	integer	Rigid wall energy calc flag
rylen	integer	Rayleigh energy calc flag
slnten	integer	Contact energy calc flag

*CONTROL_EXPLICIT_THERMAL_ALE_COUPLING

Properties for *CONTROL_EXPLICIT_THERMAL_ALE_COUPLING

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
mmset	integer	The Multi-material Set ID.
partset	integer	The Part Set ID.

*CONTROL_EXPLICIT_THERMAL_BOUNDARY

Properties for *CONTROL_EXPLICIT_THERMAL_BOUNDARY

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
lcid	integer	The Curve ID specifying Temperature vs Time.
sgset	integer	The Segment Set ID.

*CONTROL_EXPLICIT_THERMAL_CONTACT

Properties for *CONTROL_EXPLICIT_THERMAL_CONTACT

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ncycle	real	Number of cycle between checks of new contact.
partset	integer	The Part Set ID.

*CONTROL_EXPLICIT_THERMAL_INITIAL

Properties for *CONTROL_EXPLICIT_THERMAL_INITIAL

Name	Type	Description
exists	logical	true if control card exists

id	integer	If less than 0 then Element ID if greater than 0 then Set ID. Can be SOLID, SHELL, BEAM or THICK SHELL based on value of idtyp.
idtyp	integer	Type of ID. Valid values: 1-Solid, 2-Shell, 3-Beam, 4-Thick shell .
include	integer	The Include file number that the control card is in.
tempini	real	Initial Temperature.

*CONTROL_EXPLICIT_THERMAL_OUTPUT

Properties for *CONTROL_EXPLICIT_THERMAL_OUTPUT

Name	Type	Description
dtout	real/integer	Time interval between outputs. Constant float value if DTOUTYP = 0, Curve ID if DTOUTYP = 1.
dtoutyp	integer	Type of DTOUT. Valid values: 0-Constant, 1-Time vs DTOUT Curve.
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
set	integer	The Set ID. Can be SOLID, SHELL or BEAM Set based on value of setyp.
setyp	integer	Type of Set. Valid values: 1-Solid Set, 2-Shell Set, 3-Beam Set.

*CONTROL_EXPLICIT_THERMAL_PROPERTIES

Properties for *CONTROL_EXPLICIT_THERMAL_PROPERTIES

Name	Type	Description
cp	real/integer	Heat Capacity. Constant float value if CPTYP = 0, Curve ID if CPTYP = 1.
cptyp	integer	Type of CP. Valid values: 0-Constant, 1-Temperature vs Heat Capacity Curve.
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
kxx	real/integer	Heat conductivity matrix. Constant float value if respective KxxTYP = 0, Curve ID if respective KxxTYP = 1.
kxxtyp	integer	Types of Kxx. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kxy	real/integer	Heat conductivity matrix. Constant float value if respective KxyTYP = 0, Curve ID if respective KxyTYP = 1.
kxytyp	integer	Types of Kxy. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kxz	real/integer	Heat conductivity matrix. Constant float value if respective KxzTYP = 0, Curve ID if respective KxzTYP = 1.
kxztyp	integer	Types of Kxz. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kyx	real/integer	Heat conductivity matrix. Constant float value if respective KyxTYP = 0, Curve ID if respective KyxTYP = 1.
kyxtyp	integer	Types of Kyx. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kyy	real/integer	Heat conductivity matrix. Constant float value if respective KyyTYP = 0, Curve ID if respective KyyTYP = 1.
kyytyp	integer	Types of Kyy. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kyz	real/integer	Heat conductivity matrix. Constant float value if respective KyzTYP = 0, Curve ID if respective KyzTYP = 1.

kyztyp	integer	Types of Kyz. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kzx	real/integer	Heat conductivity matrix. Constant float value if respective KzxTYP = 0, Curve ID if respective KzxTYP = 1.
kzxtyp	integer	Types of Kzx. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kzy	real/integer	Heat conductivity matrix. Constant float value if respective KzyTYP = 0, Curve ID if respective KzyTYP = 1.
kzytyp	integer	Types of Kzy. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
kzz	real/integer	Heat conductivity matrix. Constant float value if respective KzzTYP = 0, Curve ID if respective KzzTYP = 1.
kzztyp	integer	Types of Kzz. Valid values: 0-Constant, 1-Temperature vs Heat Conductivity Curve.
local	integer	Flag to activate an element csys. Valid values: 0-Vecids are considered in Global csys, 1-Vecids are considered in Local Csys.
partset	integer	The Part Set ID.
vecid1	integer	The Vector ID to define x-direction.
vecid2	integer	The Vector ID to define y-direction.

*CONTROL_EXPLICIT_THERMAL_SOLVER

Properties for *CONTROL_EXPLICIT_THERMAL_SOLVER

Name	Type	Description
dtfac	real	Time step factor.
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
partset	integer	The Part Set ID.

*CONTROL_EXPLOSIVE_SHADOW

Properties for *CONTROL_EXPLOSIVE_SHADOW

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
set_option	logical	true if _SET option is present.
setid	integer	Set ID of *SET_SHELL or *SET_SOLID.

*CONTROL_FORMING_BESTFIT

Properties for *CONTROL_FORMING_BESTFIT

Name	Type	Description
exists	logical	true if control card exists
filename	string	Target mesh file in keyword format

gaponly	integer	Separation distance calculation flag
ifast	integer	Computing performance optimisation flag
ifit	integer	Best fit flag
ifset	integer	Optional flag to define a node set to be included or excluded
include	integer	The Include file number that the control card is in.
nsets	integer	An optional node set ID of three nodes from the source mesh
nsett	integer	An optional node set ID of three nodes from the target mesh
nskip	integer	Optional skipping scheme
vector	logical	true if _VECTOR option is set

*CONTROL_FORMING_INITIAL_THICKNESS

Properties for *CONTROL_FORMING_INITIAL_THICKNESS

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
lcid	integer	Load curve ID defining thickness vs distance
pid	integer	Part ID of the sheet blank
vx	real	X component of vector defining the direction of distance in load curve
vy	real	Y component of vector defining the direction of distance in load curve
vz	real	Z component of vector defining the direction of distance in load curve
x0	real	Starting position x coordinate
y0	real	Starting position y coordinate
z0	real	Starting position z coordinate

*CONTROL_FORMING_MAXID

Properties for *CONTROL_FORMING_MAXID

Name	Type	Description
exists	logical	true if control card exists
i2dynain	integer	Keyword to be output to a dynain file
include	integer	The Include file number that the control card is in.
maxide	integer	Element ID number
maxidn	integer	Node ID number
pid	integer	Part ID of the sheet blank

*CONTROL_FORMING_POSITION

Properties for *CONTROL_FORMING_POSITION

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
pid	integer	Part ID
premove	real	Distance to pre-move tool in reverse direction
target	integer	

*CONTROL_FORMING_PRE_BENDING

Properties for *CONTROL_FORMING_PRE_BENDING

Name	Type	Description
cid	integer	ID of coordinate system (only for the LOCAL option)
cx	real	X component of centre of most-bent location
cy	real	Y component of centre of most-bent location
cz	real	Z component of centre of most-bent location
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
option	integer	Keyword option
pset	integer	Part set ID
radius	real	Radius of pre-bending
vx	real	X component of axis about which blank will be bent
vy	real	Y component of axis about which blank will be bent
vz	real	Z component of axis about which blank will be bent

*CONTROL_FORMING_PROJECTION

Properties for *CONTROL_FORMING_PROJECTION

Name	Type	Description
exists	logical	true if control card exists
gap	real	Minimum gap
include	integer	The Include file number that the control card is in.
nrbst	integer	Normal direction of blank
nrtst	integer	Normal direction of tool
pidb	integer	Part id for blank
pidt	integer	Part id for tool

*CONTROL_FORMING_REMOVE_ADAPTIVE_CONSTRAINTS

Properties for *CONTROL_FORMING_REMOVE_ADAPTIVE_

CONSTRAINTS

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
pid	integer	Part id to remove adaptive constraints from

*CONTROL_FORMING_SHELL_TO_TSHELL

Properties for *CONTROL_FORMING_SHELL_TO_TSHELL

Name	Type	Description
exists	logical	true if control card exists
idsegb	integer	Set id of the segments to be generated at the bottom layer
idsegt	integer	Set id of the segments to be generated at the top layer
include	integer	The Include file number that the control card is in.
midsf	integer	Mid-plane position flag
pid	integer	Part id of the thin shell elements
thick	real	Thickness of the thick shell elements

*CONTROL_FORMING_STONING

Properties for *CONTROL_FORMING_STONING

Name	Type	Description
direct	real	Number of automatically determined directions
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
istone	integer	Stoning calculation option
itype	integer	Set type designation
length	real	Length of the stone
method	integer	Stoning method
node1	integer	Tail node defining stone moving direction
node1	integer	Head node defining stone moving direction
reverse	integer	Surface normal reversing option
sid	integer	Node/Shell set id
step	real	Stepping size of moving stone
v1	real	Vector component defining stoning direction
v2	real	Vector component defining stoning direction
v3	real	Vector component defining stoning direction
width	real	Width of the stone

*CONTROL_FORMING_STRAIN_RATIO_SMOOTH

Properties for *CONTROL_FORMING_STRAIN_RATIO_SMOOTH

Name	Type	Description
dt_cycle	real	Flag for output option (time interval or cycle number)
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
weight	real	Coefficient in equation

*CONTROL_FORMING_TEMPLATE

Properties for *CONTROL_FORMING_TEMPLATE

Name	Type	Description
al_fe	string	A=Aluminium blank, F=steel
amax	real	Maximum allowable acceleration
blkid	integer	Part (stype=0) or part set (stype=1) ID that defines the blank
bndl	integer	Part that defines the lower binder
bndu	integer	Part that defines the upper binder
d3plt	integer	Number of output states in the D3PLOT database
density	real	Density
dieid	integer	Part that defines the die
e	real	Youngs modulus
exists	logical	true if control card exists
fs	real	Friction coefficient
gap	real	Home gap between rigid tools
idtemp	integer	Type of forming process
include	integer	The Include file number that the control card is in.
k	real	Strength coefficient for exponential hardening
lcss	integer	Loadcurve for stress-strain relationship
lvlada	integer	Maximum adaptive level
mtyp	integer	Material type
n	real	Exponent for exponential hardening
patern	integer	Velocity profile of moving tool
pnch	integer	Part that defines the punch
pr	real	Poissons ratio
prebd	real	Distance between lower binder and punch
r00	real	Material anisotropic parameter R00
r45	real	Material anisotropic parameter R45
r90	real	Material anisotropic parameter R90

sizeada	real	Minimum element size permitted in the adaptive mesh
stype	integer	0->blkid is PART, 1->PARTSET NOTE don’t use <type> as in stat_header
thick	real	Blank thickness
timsada	integer	Total number of adaptive steps during the forming simulation
unit	integer	Units for simulation
vid	integer	Vector ID defining direction of movement
vmax	real	Maximum allowable tool velocity
vx	real	X vector component of movement of punch
vy	real	Y vector component of movement of punch
vz	real	Z vector component of movement of punch

*CONTROL_FORMING_TOLERANC

Properties for *CONTROL_FORMING_TOLERANC

Name	Type	Description
dt_cycle	real	Flag for output option (time interval or cycle number)
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
output	integer	Output Flag
weight	real	Coefficient in equation

*CONTROL_FORMING_TRAVEL

Properties for *CONTROL_FORMING_TRAVEL

Name	Type	Description
exists	logical	true if control card exists
follow	integer	Part for tool to follow
gap	real	Min distance between tool and target in the home position
include	integer	The Include file number that the control card is in.
phase	integer	Phase number
pid	integer	Part ID of tool
target	integer	Move tool PID to meet part TARGET
travel	real	Distance to move tool along VID
vid	integer	Vector ID defining direction of travel

*CONTROL_FORMING_TRIM_MERGE

Properties for *CONTROL_FORMING_TRIM_MERGE

Name	Type	Description
exists	logical	true if control card exists
gapm	real	Gap distance between two open ends of a trim loop curve in the model
imerge	integer	Activation flag
include	integer	The Include file number that the control card is in.

*CONTROL_FORMING_TRIM_SOLID_REFINEMENT

Properties for *CONTROL_FORMING_TRIM_SOLID_REFINEMENT

Name	Type	Description
exists	logical	true if control card exists
ilevel	integer	Adaptive refinement level
include	integer	The Include file number that the control card is in.
irefine	integer	Flag to activate trimming of a multi-layer sandwiched part

*CONTROL_FORMING_UNFLANGING

Properties for *CONTROL_FORMING_UNFLANGING

Name	Type	Description
charlen	real	Max flange height
dist	real	Distance tolerance for auto-SPC along flange roots
dvid	integer	Not used
epsmx	real	Max effective plastic strain, beyond which elements are deleted
exists	logical	true if control card exists
iflimit	integer	Iteration limit for first phase of unfolding
ilinear	integer	Unfolding algorithm selection flag
include	integer	The Include file number that the control card is in.
nb1	integer	Start node ID on a flange root boundary
nb2	integer	ID of a node in the middl of the flange root boundary
nb3	integer	End node ID on a flange root boundary
ndouter	integer	A node ID on the outer flange boundary
noption	integer	Flag to turn on unfolding simulation
nunbend	integer	Estimated number of unbending
output	logical	TRUE if _<OPTION> is OUTPUT.
stfbend	real	Unflanging stiffness
stfcnt	real	Normal stiffness
thmn	real	Min thickness below which elements are deleted
thmx	real	Max thickness beyond which elements are deleted

*CONTROL_FORMING_USER

Properties for *CONTROL_FORMING_USER

Name	Type	Description
adatims	integer	Total number of adaptive steps during the forming simulation
al_fe	string	A=Aluminium blank, F=steel
amax	real	Maximum allowable acceleration
blank	integer	Part (stype=0) or part set (stype=1) ID for blank
d3plot	integer	Number of output states in the D3PLOT database
density	real	Density
e	real	Youngs modulus
exists	logical	true if control card exists
fs	real	Friction coefficient
gap	real	Minimum gap between tools
include	integer	The Include file number that the control card is in.
k	real	Strength coefficient for exponential hardening
lcss	integer	Loadcurve for stress-strain relationship
lvlada	integer	Maximum adaptive level
mtype	integer	Material type
n	real	Exponent for exponential hardening
patern	integer	Velocity profile of moving tool
pr	real	Poissons ratio
r00	real	Material anisotropic parameter R00
r45	real	Material anisotropic parameter R45
r90	real	Material anisotropic parameter R90
sizeada	real	Minimum element size permitted in the adaptive mesh
stype	integer	Flag for part/part set
thick	real	Blank thickness
unit	integer	Units for simulation
vmax	real	Maximum allowable tool velocity

*CONTROL_FREQUENCY_DOMAIN

Properties for *CONTROL_FREQUENCY_DOMAIN

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
mpn	real	Large mass added per node.
refgeo	integer	Flag for reference geometry in acoustic eigenvalue analysis

*CONTROL_FREQUENCY_RESPONSE_FUNCTION

Properties for *CONTROL_FREQUENCY_RESPONSE_FUNCTION

Name	Type	Description
dampf	real	Modal damping coefficient
dmpmas	real	Mass proportional damping constant in Rayleigh damping
dmpstf	real	Stiffness proportional damping constant in Rayleigh damping
dof1	integer	Applicable degrees-of-freedom for excitation input
dof2	integer	Applicable degrees-of-freedom for response output
exists	logical	true if control card exists
fmax	real	Maximum frequency for FRF output
fmin	real	Minimum frequency for FRF output
fnmax	real	Optional maximum natural frequency
include	integer	The Include file number that the control card is in.
lcdam	integer	Loadcurve ID defining modal damping coefficient
lctyp	integer	Type of load curve
mdmax	integer	Last mode employed in FRF computation
mdmin	integer	First mode employed in FRF computation
n1	integer	Node (n1typ=0) / node set (n1typ=1) /segment set (n1typ=2) ID for excitation input
n1typ	integer	Type of N1
n2	integer	Node (n2typ=0) /node set (n2typ=1) /segment set (n2typ=2) ID for response output
n2typ	integer	Type of N2
nfreq	integer	Number of frequencies for FRF output
restrt	integer	Restart option
vad1	integer	Excitation input type
vad2	integer	Response output type
vid	integer	Vector ID for DOF1=4

*CONTROL_HOURGLASS

Properties for *CONTROL_HOURGLASS

Name	Type	Description
exists	logical	true if control card exists
f_936	integer	Internal flag to set 936 compatibility
ihq	integer	Hourglass viscosity type
include	integer	The Include file number that the control card is in.
qh	real	Default hourglass coefficient

*CONTROL_IMPLICIT_AUTO

Properties for *CONTROL_IMPLICIT_AUTO

Name	Type	Description
dtexp	real	time in explicit before switch
dtmax	integer	Maximum allowable timestep. Loadcurve if negative
dtmin	real	Minimum allowable timestep
exists	logical	true if control card exists
hcmax	integer	Mid-point relative Euclidian residual norm max tolerance
hcmin	integer	Mid-point relative Euclidian residual norm min tolerance
hmmax	integer	Mid-point relative maximum residual norm max tolerance
hmmin	integer	Mid-point relative maximum residual norm min tolerance
hnrmax	integer	Mid-point absolute Nodal Rotational norm tolerance
hntmax	integer	Mid-point absolute Nodal Translational norm tolerance
hrrmax	integer	Mid-point absolute Rigid body Rotational norm tolerance
hrtmax	integer	Mid-point absolute Rigid body Translational norm tolerance
iauto	integer	Automatic timestep control flag. Loadcurve if negative
include	integer	The Include file number that the control card is in.
iteopt	integer	Optimum equilibrium iteration count per timestep
itewin	integer	Allowable iteration window (no. of iterations)
kcycle	integer	number of explicit cycles before switch
kfail	integer	number of failed implicit attempts before switch

*CONTROL_IMPLICIT_BUCKLE

Properties for *CONTROL_IMPLICIT_BUCKLE

Name	Type	Description
bckmth	integer	Method to extract buckling modes
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
nmode	integer	number of buckling modes to calculate

*CONTROL_IMPLICIT_CONSISTENT_MASS

Properties for *CONTROL_IMPLICIT_CONSISTENT_MASS

Name	Type	Description
exists	logical	true if control card exists
iflag	integer	Consistent mass matrix flag

include	integer	The Include file number that the control card is in.

*CONTROL_IMPLICIT_DYNAMICS

Properties for *CONTROL_IMPLICIT_DYNAMICS

Name	Type	Description
alpha	real	Composite time integration constant
beta	real	Newmark time integration constant
exists	logical	true if control card exists
gamma	real	Newmark time integration constant
imass	integer	Implicit analysis type
include	integer	The Include file number that the control card is in.
irate	integer	rate effect switch
tdybir	integer	birth time for dynamic terms. Loadcurve if negative
tdybur	real	burial
tdydth	real	death

*CONTROL_IMPLICIT_EIGENVALUE

Properties for *CONTROL_IMPLICIT_EIGENVALUE

Name	Type	Description
center	real	Centre frequency
eigmth	integer	Eigenvalue extraction method
evdump	integer	Flag for writing eigenvalues and eigenvectors
exists	logical	true if control card exists
ibeam	integer	Beam element formulation for implicit
include	integer	The Include file number that the control card is in.
iparm1	integer	Minimum block size for the Cholesky factorization (for eigmth=101) or Maximum number of iterations (for eigmth=102)
iparm2	integer	Maximum block size for the Cholesky factorization (for eigmth=101) or Block size (for eigmth=102)
iparm3	integer	Node set ID
iparm4	integer	MCMS minimum group/substructure size
ishell	integer	Shell element formulation for implicit
isolid	integer	Solid element formulation for implicit
itshell	integer	Thick shell element formulation for implicit
lflag	integer	Left end point finite flag
lftend	real	Left end point of interval
mstres	integer	stress compute flag

mstrscl	real	Scaling for computing velocity
neig	integer	#eigenvalues to extract; loadcurve if negative
rflag	integer	Right end point finite flag
rhtend	real	Right end point of interval
rparm1	real	Eigenvalue expansion factor (for eigmth=101) or Convergence tolerance (for eigmth=102)
rparm2	real	BLR preconditioner tolerance
shfscl	real	Shift scale

*CONTROL_IMPLICIT_EXPLICIT_HYBRID

Properties for *CONTROL_IMPLICIT_EXPLICIT_HYBRID

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
psid	integer	Part set ID

*CONTROL_IMPLICIT_FORMING

Properties for *CONTROL_IMPLICIT_FORMING

Name	Type	Description
birth	real	birth time
death	real	death time
dt0	real	initial time step size
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ioption	integer	1:gravity 2:binder
nsmax	integer	max number of implicit steps
nsmin	integer	min number of implicit steps
penchk	real	penetration allowed as ratio of part thickness

*CONTROL_IMPLICIT_GENERAL

Properties for *CONTROL_IMPLICIT_GENERAL

Name	Type	Description
cnstn	integer	Consistent tangent stiffness flag
dt0	real	Initial timestep for implicit analysis
exists	logical	true if control card exists
form	integer	Element formulation to use.

igs	integer	Geometric (initial stress) stiffness flag
imflag	integer	Implicit/explicit switching flag; loadcurve if negative
imform	integer	Element formulation switching flag
include	integer	The Include file number that the control card is in.
nsbs	integer	Number of steps in non-linear springback
zero_v	integer	flag to zero vels before switch to implicit

*CONTROL_IMPLICIT_INERTIA_RELIEF

Properties for *CONTROL_IMPLICIT_INERTIA_RELIEF

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ircnt	integer	Lowest IRCNT modes
irflag	integer	Inertia relief flag
thresh	real	Threshold for rigid body node

*CONTROL_IMPLICIT_JOINTS

Properties for *CONTROL_IMPLICIT_JOINTS

Name	Type	Description
exists	logical	true if control card exists
icylin	integer	Treatment of cylindrical joints
include	integer	The Include file number that the control card is in.
irevol	integer	Treatment of revolute joints
ispher	integer	Treatment of spherical joints

*CONTROL_IMPLICIT_MODAL_DYNAMIC

Properties for *CONTROL_IMPLICIT_MODAL_DYNAMIC

Name	Type	Description
dtout	real	Modal dynamics output interval
exists	logical	true if control card exists
filename2	string	Constraint modes file name
filename	string	Eigen modes file name
include	integer	The Include file number that the control card is in.
integ	integer	Integration method
md_strs	integer	Modal dynamic stress flag

mdflag	integer	Modal dynamic flag
nsid	integer	Node set ID of the nodes in the modal model that are subjected to loads
zeta	real	Modal dynamic damping constant

*CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING

Member functions

- GetCoefficient(index*[integer]*)
- RemoveCoefficient(index*[integer]*)
- SetCoefficient(index*[integer]*, mode/frequency*[integer/real]*, zeta*[real]*)

Properties for *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING

Name	Type	Description
coefficients	integer	Number of coefficients for SPECIFIC and FREQUENCY_RANGE options
exists	logical	true if control card exists
frequency_range	boolean	If FREQUENCY_RANGE option is used
include	integer	The Include file number that the control card is in.
specific	boolean	If SPECIFIC option is used
zeta	real	Modal dynamic damping constant

Details of functions

GetCoefficient(index*[integer]*)

Description

Returns the damping coefficient data for an index in *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING.

Arguments

Name	Type	Description
index	integer	The index you want the data for. **Note that indices start at 0, not 1**.

Return type

An array containing the mode id/frequency and damping coefficient values.

Example

To get the damping data for the 3rd index for *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING in model m:

```
if (m.control.implicit_modal_dynamic_damping.coefficients >= 3)
{
    var data = m.control.implicit_modal_dynamic_damping.GetCoefficient(2);
}
```

RemoveCoefficient(index*[integer]*)

Description

Removes the damping coefficient data for an index in *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING.

Arguments

Name	Type	Description
index	integer	The index you want to delete damping data for. **Note that indices start at 0, not 1**.

Return type

No return value.

Example

To delete the damping data for the 3rd index for *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING in model m:

```
if (m.control.implicit_modal_dynamic_damping.coefficients >= 3)
{
    m.control.implicit_modal_dynamic_damping.RemoveCoefficient(2);
}
```

SetCoefficient(index*[integer]*, mode/frequency*[integer/real]*, zeta*[real]*)

Description

Sets the damping coefficient data for an index in *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING.

Arguments

Name	Type	Description
index	integer	The index you want to set the data for. **Note that indices start at 0, not 1**.
mode/frequency	integer/real	The mode ID (_SPECIFIC) or frequency (_FREQUENCY_RANGE).
zeta	real	Damping coefficient

Return type

No return value.

Example

To set the damping data for the 3rd index for *CONTROL_IMPLICIT_MODAL_DYNAMIC_DAMPING_SPECIFIC in model m to have mode ID 10 and damping coefficient 0.1:

```
m.control.implicit_modal_dynamic_damping.SetCoefficient(2, 10, 0.1);
```

---

*CONTROL_IMPLICIT_MODES

Properties for *CONTROL_IMPLICIT_MODES

Name	Type	Description
exists	logical	true if control card exists
ibase	integer	Offset for numbering
id3mode	integer	Write d3mode file flag
include	integer	The Include file number that the control card is in.
iresvec	integer	Converting the attachment modes to residual vectors flag
istress	integer	Flag to compute stresses
neig	integer	Number of eigenmodes
nsida	integer	node set for attachment modes

nsidc	integer	node set constraint modes
opt	integer	Can be <BLANK> or _BINARY
se_damp	string	Name of superelement damping matrix
se_filename	string	File name
se_inert	string	Name of superelement inertia matrix
se_mass	string	Name of superelement mass matrix
se_stiff	string	Name of superelement stiffness matrix

*CONTROL_IMPLICIT_ORDERING

Properties for *CONTROL_IMPLICIT_ORDERING

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
nmetis	integer	Number of times to use Metis
order	integer	Ordering option

*CONTROL_IMPLICIT_RESIDUAL_VECTOR

Properties for *CONTROL_IMPLICIT_RESIDUAL_VECTOR

Name	Type	Description
exists	logical	true if control card exists
iformat	integer	Format for processing eigenmodes
include	integer	The Include file number that the control card is in.
iresvec	integer	Residual vector control flag
neig	integer	Number of eigenmodes to compute for the purpose of orthogonalizing the computed load

*CONTROL_IMPLICIT_SOLUTION

Properties for *CONTROL_IMPLICIT_SOLUTION

Name	Type	Description
abstol	real	absolute convergence tol
arcalf	integer	relative influence predictor step
arcctl	integer	Arc length controlling node ID
arcdir	integer	Arc length controlling node direction
arcdmp	integer	Arc length damping option
arclen	real	Arc length size
arcmth	integer	Arc length method

arcpsi	integer	relative influence load/time parameter
arctim	integer	initiation time
awgt	real	weight factor
cpchk	integer	Contact penetration check flag
d3itctl	integer	D3ITER database control
dctol	real	Displacement convergence tolerance
diverg	integer	Divergence flag
dmtol	real	Maximum displacement convergence tolerance
dnorm	integer	Displacement norm for convergence test
ectol	real	Energy convergence tolerance
emtol	real	Maximum energy convergence tolerance
exists	logical	true if control card exists
ilimit	integer	Iteration limit between automatic stiffness reformations
include	integer	The Include file number that the control card is in.
irad	real	curve factor
istif	integer	Initial stiffness formulation flag
lsdir	integer	search direction
lsmtd	integer	search method
lstol	real	Line search convergence tolerance
maxref	integer	Stiffness reformation limit per time step
nlnorm	real	non-linear convergence type
nlprint	integer	non-linear solver print flag
nrtol	real	Nodal rotational convergence tolerance
nsolvr	integer	Non-linear equation solver method
nttol	real	Nodal translational convergence tolerance
rctol	real	Residual (force) convergence tolerance
rmtol	real	Maximum residual convergence tolerance
rrtol	real	Rigid body rotational convergence tolerance
rttol	real	Rigid body translational convergence tolerance
srad	real	radius of influence
sred	real	step reduction factor

*CONTROL_IMPLICIT_SOLVER

Properties for *CONTROL_IMPLICIT_SOLVER

Name	Type	Description
autospc	integer	AUTOSPC switch
autotol	real	AUTOSPC tolerance
drcm	integer	Drilling rotation constraint method

drcprm	real	Drilling rotation constraint parameter
emxdmp	integer	Flag for dumping elemental stiffness and mass matrices
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
iparm1	integer	Maximum number of iterations
lcpack	integer	Matrix assembly package
lprint	integer	Linear solver print flag
lsolvr	integer	Linear equation solver method
mtxdmp	integer	flag to dump matrix
negev	integer	Negative eigenvalue flag
order	integer	Ordering option
rdcmem	integer	Factor for capping the amount of dynamic memory requested
rparm1	integer	Absolute tolerance for convergence
rparm2	integer	Relative tolerance for convergence

*CONTROL_IMPLICIT_STABILIZATION

Properties for *CONTROL_IMPLICIT_STABILIZATION

Name	Type	Description
exists	logical	true if control card exists
ias	integer	Artificial stabilization flag
include	integer	The Include file number that the control card is in.
scale	integer	scale factor for artificial stabilization. Loadcurve if negative
tend	real	End time
tstart	real	Start time

*CONTROL_IMPLICIT_STATIC_CONDENSATION

Properties for *CONTROL_IMPLICIT_STATIC_CONDENSATION

Name	Type	Description
binary	integer	flag to set _BINARY option
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
sc_flag	integer	Static condensation control flag
sc_nsid	integer	Node set ID for nodes to be preserved in the procedure
sc_psid	integer	Part set ID for parts to be included in the procedure
se_filename	string	File name
se_inert	string	Name of superelement inertia matrix

se_mass	string	Name of superelement mass matrix
se_stiff	string	Name of superelement stiffness matrix

*CONTROL_IMPLICIT_TERMINATION

Properties for *CONTROL_IMPLICIT_TERMINATION

Name	Type	Description
absol	real	Terminate based on absolute total displacement in the Euclidean norm.
delta1	real	Terminate based on rel total displacement in max norm
deltau	real	Terminate based on rel total displacement in Euclidean norm
exists	logical	true if control card exists
ietol	real	Terminate based on internal energy
include	integer	The Include file number that the control card is in.
ketol	real	Terminate based on kinetic energy
nstep	integer	Consecutive implicit time steps
tetol	real	Terminate based on total energy

*CONTROL_MPP_CONTACT_GROUPABLE

Properties for *CONTROL_MPP_CONTACT_GROUPABLE

Name	Type	Description
exists	logical	true if control card exists
grp	integer	GROUPABLE algorithm options
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_AUTOMATIC

Properties for *CONTROL_MPP_DECOMPOSITION_AUTOMATIC

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_BAGREF

Properties for *CONTROL_MPP_DECOMPOSITION_BAGREF

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_CHECK_SPEED

Properties for *CONTROL_MPP_DECOMPOSITION_CHECK_SPEED

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_CONTACT_ISOLATE

Properties for *CONTROL_MPP_DECOMPOSITION_CONTACT_ISOLATE

Name	Type	Description
exists	logical	true if control card exists
id1	integer	Contact ID 1 to distrubute
id2	integer	Contact ID 2 to distrubute
id3	integer	Contact ID 3 to distrubute
id4	integer	Contact ID 4 to distrubute
id5	integer	Contact ID 5 to distrubute
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_DISABLE_UNREF_CURVES

Properties for *CONTROL_MPP_DECOMPOSITION_DISABLE_UNREF_CURVES

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_ALE_ELEMENTS

Properties for *CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_ALE_ELEMENTS

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
overlap	logical	Decompose the structure and ALE domains together?

*CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_SPH_ELEMENTS

Properties for *CONTROL_MPP_DECOMPOSITION_DISTRIBUTE_SPH_ELEMENTS

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_ELCOST

Properties for *CONTROL_MPP_DECOMPOSITION_ELCOST

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
itype	integer	Hardware specific cost profile

*CONTROL_MPP_DECOMPOSITION_FILE

Properties for *CONTROL_MPP_DECOMPOSITION_FILE

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
name	string	decomposition file

*CONTROL_MPP_DECOMPOSITION_FLAG_STRESS_STRAIN_CURVE

Properties for *CONTROL_MPP_DECOMPOSITION_FLAG_STRESS_STRAIN_CURVE

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_METHOD

Properties for *CONTROL_MPP_DECOMPOSITION_METHOD

Name	Type	Description

exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
name	string	decomposition method

*CONTROL_MPP_DECOMPOSITION_NUMPROC

Properties for *CONTROL_MPP_DECOMPOSITION_NUMPROC

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
n	integer	number of processors

*CONTROL_MPP_DECOMPOSITION_OUTDECOMP

Properties for *CONTROL_MPP_DECOMPOSITION_OUTDECOMP

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
itype	integer	Database format

*CONTROL_MPP_DECOMPOSITION_RCBLOG

Properties for *CONTROL_MPP_DECOMPOSITION_RCBLOG

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
name	string	decomposition file

*CONTROL_MPP_DECOMPOSITION_ REDECOMPOSITION

Properties for *CONTROL_MPP_DECOMPOSITION_REDECOMPOSITION

Name	Type	Description
defgeo	integer	Geometry for decomposition
exists	logical	true if control card exists
freq	real	Time interval between redecomposition
include	integer	The Include file number that the control card is in.
remsph	integer	Flag to remove deactivated SPH particles

sampt	real	Time interval for collecting element cost profile to use in the next REDECOMP step.
stime	real	Start time for redecomposition
weight	real	Element cost scale factor for element in contact

*CONTROL_MPP_DECOMPOSITION_SCALE_CONTACT_COST

Properties for *CONTROL_MPP_DECOMPOSITION_SCALE_CONTACT_COST

Name	Type	Description
exists	logical	true if control card exists
id10	integer	Contact ID 10 to distrubute
id11	integer	Contact ID 11 to distrubute
id12	integer	Contact ID 12 to distrubute
id13	integer	Contact ID 13 to distrubute
id14	integer	Contact ID 14 to distrubute
id15	integer	Contact ID 15 to distrubute
id1	integer	Contact ID 1 to distrubute
id2	integer	Contact ID 2 to distrubute
id3	integer	Contact ID 3 to distrubute
id4	integer	Contact ID 4 to distrubute
id5	integer	Contact ID 5 to distrubute
id6	integer	Contact ID 6 to distrubute
id7	integer	Contact ID 7 to distrubute
id8	integer	Contact ID 8 to distrubute
id9	integer	Contact ID 9 to distrubute
include	integer	The Include file number that the control card is in.
sf	real	Scale factor

*CONTROL_MPP_DECOMPOSITION_SCALE_FACTOR_SPH

Properties for *CONTROL_MPP_DECOMPOSITION_SCALE_FACTOR_SPH

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
sf	real	Scale factor

*CONTROL_MPP_DECOMPOSITION_SHOW

Properties for *CONTROL_MPP_DECOMPOSITION_SHOW

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_DECOMPOSITION_TRANSFORMATION

Member functions

- GetTransformation(row*[integer]*)
- RemoveTransformation(row*[integer]*)
- SetTransformation(row*[integer]*, type or array*[string or array]*, V1*[real]*, V2 to V9 (optional)*[real]*)

Properties for *CONTROL_MPP_DECOMPOSITION_TRANSFORMATION

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
rows	integer	Number of rows.

Details of functions

GetTransformation(row*[integer]*)

Description

Returns the transformation for a row in *CONTROL_MPP_DECOMPOSITION_TRANSFORMATION.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note that rows start at 0, not 1**.

Return type

An array containing the transformation type and the parameters V1 to V9.

Example

To get the transformation for the 3rd row for *CONTROL_MPP_DECOMPOSITION_TRANSFORMATION in model m:

```
if(m.control.mpp_decomposition_transformation.rows >= 3)
{
    var type = m.control.mpp_decomposition_transformation.GetTransformation(2);
}
```

RemoveTransformation(row*[integer]*)

Description

Removes a row in *CONTROL_MPP_DECOMPOSITION_TRANSFORMATION.

Arguments

Name	Type	Description
row	integer	The row number to be deleted. **Note that rows start at 0, not 1**. If there are rows under this one, they will be shifted up.

Return type

No return value.

Example

To delete the 3rd row for *CONTROL_MPP_DECOMPOSITION_TRANSFORMATION in model m:

```
if(m.control.mpp_decomposition_transformation.rows >= 3)
{
    m.control.mpp_decomposition_transformation.RemoveTransformation(2);
}
```

SetTransformation(row*[integer]*, type or array*[string or array]*, V1*[real]*, V2 to V9 (optional)*[real]*)

Description

Sets the type and the parameters V1 to V9 for a row in *CONTROL_MPP_DECOMPOSITION_ TRANSFORMATION. Note: If the row already exists, the type and the parameters will be overwritten.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that rows start at 0, not 1**.
type or array	string or array	The string repesenting the type of the transformation. Alternatively an array with this and the following parameters on it.
V1	real	The float repesenting the parameter V1
V2 to V9 (optional)	real	The floats repesenting the parameters V2 to V9. These are only used for VEC3, C2R, S2R and MAT.

Return type

No return value.

Example

To set in the first row a transformation of type to RZ with V1=1.2 for *CONTROL_MPP_DECOMPOSITION_ TRANSFORMATION in model m:

```
m.control.implicit_modal_dynamic_damping.SetTransformation(0, "RZ", 1.2);
```

or

```
var a = new Array("RZ", 1.2);
      m.control.implicit_modal_dynamic_damping.SetTransformation(0, a);
```

To set in the third row a transformation of type to VEC3 with V1=1.0, V2=2.0... V9=9.0 for *CONTROL_MPP_ DECOMPOSITION_TRANSFORMATION in model m:

```
m.control.implicit_modal_dynamic_damping.SetTransformation(2, "VEC3", 1, 2, 3,
4, 5, 6, 7, 8, 9);
```

or

```
var a = new Array("VEC3", 1, 2, 3, 4, 5, 6, 7, 8, 9);
      m.control.implicit_modal_dynamic_damping.SetTransformation(2, a);
```

---

*CONTROL_MPP_IO_BINOUTONLY

Properties for *CONTROL_MPP_IO_BINOUTONLY

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_LSTC_REDUCE

Properties for *CONTROL_MPP_IO_LSTC_REDUCE

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_NOD3DUMP

Properties for *CONTROL_MPP_IO_NOD3DUMP

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_NODUMP

Properties for *CONTROL_MPP_IO_NODUMP

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_NOFAIL

Properties for *CONTROL_MPP_IO_NOFAIL

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_NOFULL

Properties for *CONTROL_MPP_IO_NOFULL

Name	Type	Description

exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_IO_SWAPBYTES

Properties for *CONTROL_MPP_IO_SWAPBYTES

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_MATERIAL_MODEL_DRIVER

Properties for *CONTROL_MPP_MATERIAL_MODEL_DRIVER

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_MPP_REBALANCE

Properties for *CONTROL_MPP_REBALANCE

Name	Type	Description
exists	logical	true if control card exists
icoor	integer	Coordinates used in rebalance
icost	integer	Element costs used in rebalance
include	integer	The Include file number that the control card is in.
ncycle	integer	Number of cycles between rebalance steps
thres	real	Percent threshold for rebalancing

*CONTROL_NONLOCAL

Properties for *CONTROL_NONLOCAL

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
mem	integer	%age increase in memory for *MAT_NONLOCAL usage

*CONTROL_OUTPUT

Properties for *CONTROL_OUTPUT

Name	Type	Description
cdetol	real	Tolerance for output of *DEFINE_CURVE discretization warnings
demden	integer	Output DEM density data to d3plot database
engout	integer	Flag to output contact sliding energy densities for mortar contact
eocs	integer	Elout coordinate system option
exists	logical	true if control card exists
frfreq	integer	Output frequency for failed element report
gmdt	real	output interval for *INTERFACE_SSI_AUX
hisnout	integer	Flag to invoke output of extra history variable names
iaccop	integer	Flag for accels in d3thdt to be averaged
ibsf	integer	Flag to invoke output of *SET_BEAM data
icrfile	integer	Output node and element sets used in computing secforc data
ierode	integer	output eroded energy
iflush	integer	i/o buffer flushing interval (t-steps)
ikedit	integer	Status report interval to d3hsp
include	integer	The Include file number that the control card is in.
insf	integer	Flag to invoke output of *SET_NODE data
ip1dblt	integer	output of 1D seatbelt created for 2D seatbelt to sbtout
ipcurv	integer	output curve data flag
ipnint	integer	Flag to print initial timesteps at cycle #1
iprtf	integer	Print flag for RBDOUT and MATSUM files
isolsf	integer	Flag to invoke output of *SET_SOLID data
issf	integer	Flag to invoke output of *SET_SHELL data
minfo	integer	Output penetration information
mlkbag	integer	Flag to invoke output of accumulated airbag mass leakage in ABSTAT
msgflg	integer	Option for printing detail message to d3msg
msgmax	integer	max num messags
neecho	integer	Print suppression during input: echo file
newleg	integer	New legends
npopt	integer	Print suppression during input: printer file
nrefup	integer	Flag to update individual beam 3rd nodes
opifs	real	Output interval for interface file
penout	integer	Flag to output contact penetration for mortar contact
phschng	integer	Message to messag file for phase change on materials 216, 217 and 218
shlsig	integer	Flag to extrapolate stresses for shells with 8 integration points to nodes
solsig	integer	Flag to extrapolate stresses/history variables

spc2bnd	integer	Flag to convert constraints on rigid bodies to equivalent *BOUNDARY_PRESCRIBED_ MOTION_RIGID motion
tet10s8	integer	tet connectivity output
tolev	integer	Timing output levels

*CONTROL_PARALLEL

Properties for *CONTROL_PARALLEL

Name	Type	Description
consty	integer	Consistency (Accuracy) flag
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ncpu	integer	#cpus to use
numrhs	integer	#rh sides written
para	integer	Flag for parallel force assembly

*CONTROL_PORE_AIR

Properties for *CONTROL_PORE_AIR

Name	Type	Description
air_p	real	Pressure of atmospheric air
air_ro	real	Density of atmospheric air
anamsg	integer	Flag to turn off printing of pore air analysis status message
eterm	real	Event termination time
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_PORE_FLUID

Properties for *CONTROL_PORE_FLUID

Name	Type	Description
atype	integer	Analysis type
conmax	real	damping factor
conv	real	conduction factor
datum	real	Z elevation of datum
eterm	real	event time termination
etflag	integer	Flag for interpretation of time
exists	logical	true if control card exists
fmax	real	max seepage factor

fmin	real	min seepage factor
ftied	real	Analysis type
grav	real	Gravitational acceleration for Ro.g.h
include	integer	The Include file number that the control card is in.
output	integer	Output flag for stresses
pf_bulk	real	Default bulk modulus of pore fluid
pf_rho	real	Default pore water density
targ	real	target for change of excess pressure
therm	real	thermal vol expansion coeff
tmf	integer	Time magnification factor on seepage. Loadcurve if negative
wtable	real	Default elevation of water table

*CONTROL_PWP_AUTO_TMF

Properties for *CONTROL_PWP_AUTO_TMF

Name	Type	Description
dpwmax	real	Max rate of change of pwp water head (m/s)
exists	logical	true if control card exists
fmax	real	Maximum factor on seepage calc
fmin	real	Minimum factor on seepage calc
include	integer	The Include file number that the control card is in.
sprfac	real	factor for reducing feedback
targ	real	Target max pwp change/thermal timestep

*CONTROL_PZELECTRIC

Properties for *CONTROL_PZELECTRIC

Name	Type	Description
abstol	real	Absolute convergence tolerance
epzmsg	interger	Flag to determine if electric flux and electric field at the element center of piezoelectric material is outputed to d3plot
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
maxitr	integer	Maximum number of iterations 2
msgitr	integer	Output iteration message level
ndtrfk	real	Reform the dielectric stiffness matrix every NDTRFK time steps
reltol	real	Relative convergence tolerance
solver	integer	Piezoelectric solver type

*CONTROL_REFERENCE_CONFIGURATION

Properties for *CONTROL_REFERENCE_CONFIGURATION

Name	Type	Description
exists	logical	true if control card exists
include	integer	the Include file number that the control card is in.
iter	integer	iter keyword option
iterfile	string	base name of two files for the ITER keyword option
maxiter	integer	max number of iterations
method	integer	iterative method
step	real	step size used in iterations
targetfile	string	file containing all nodes of the target geometry
tol	real	tolerance used to determine convergence

*CONTROL_REMESHING

Properties for *CONTROL_REMESHING

Name	Type	Description
cid	integer	coordinate system id
dtmin	real	timestep size for remesh
efg	integer	efg keyword option
exists	logical	true if control card exists
iaat	integer	interactive adaptivity adjustable tolerance
iat	integer	interactive adaptivity
iat1	real	tolerance of shear distortion indicator for interactive adaptivity
iat2	real	tolerance of unbalanced nodal distribution indicator for interactive adaptivity
iat3	real	tolerance of volumetric change indicator for interactive adaptivity
icurv	integer	number of elements along radius
ier	integer	remeshing with element erosion
include	integer	The Include file number that the control card is in.
ivt	integer	internal variable transfer in adaptive EFG
mfrac	real	mass ratio gain required for remesh
mm	integer	monotonic mesh resizing
rmax	real	Maximum edge length
rmin	real	Minimum edge length
segang	real	angular mesh size in 3-D axisymmetric remeshing
vfloss	real	necessary VF loss for remesh

*CONTROL_RIGID

Properties for *CONTROL_RIGID

Name	Type	Description
exists	logical	true if control card exists
gjadstf	real	Joint rotational stiffness
gjadvsc	real	Joint rotational damping
include	integer	The Include file number that the control card is in.
jntf	integer	Generalized joint stiffness formulation
lmf	integer	Switch explicit/implicit joint formulation
metalf	integer	metalforming option
norbic	integer	Circumvent rigid body inertia check
orthmd	integer	Orthogonalise modes wrt each other
partm	integer	Use global mass matrix for mass distribution
plotel	integer	Automatic generation of *ELEMENT_PLOTEL
rbsms	integer	Flag to apply consistent treatment of rigid bodies in selective mass scaling
sparse	integer	Use sparse xply routines for modal & stiffness damping matrices
tjadstf	real	Joint translational stiffness
tjadvsc	real	Joint translational damping

*CONTROL_SHELL

Properties for *CONTROL_SHELL

Name	Type	Description
bwc	integer	Warping stiffness flag for Belytschko-Tsay shells
cntco	integer	include shell ref surface offset
cstyp6	integer	Coord sys for type 6 element
delfr	integer	delete shells where neighbours fail
drcmth	integer	drilling rotation constraint method.
drcpsid	integer	part set for drilling rotation constraint method.
drcpsrm	real	drilling rotation constraint parameter.
esort	integer	Degenerate shell sorting flag (was ITRIST)
excl	integer	.eq.1 if excl above
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
intgrd	integer	Gauss/Lobatto intg rule switch
intperr	integer	Flag for behavior in case of unwanted interp. or extrap. of initial stresses
irnxx	integer	Hughes-Liu shell normal update option

irquad	integer	intg rule
istupd	integer	Shell thickness change option
itsflg	integer	initial transverse shear stress
keepcs	integer	keep contact segs of failed shells
lamsht	integer	Laminated shell theory update flag
lispsid	integer	Part set ID related to *INITIAL_STRESS_SHELL.
miter	integer	Plane stress plasticity option
nfail1	integer	Flag for distorted 1 intg point shell check
nfail4	integer	Flag for distorted 4 intg point shell check
proj	integer	Projection method for warping stiffness
psnfail	integer	part set id for check
psstupd	integer	part set for thichness update, -ve to exclude
rotascl	real	Scale factor for rotary shell mass
sidt4tu	integer	part set for type 4 thickness update where elastic strains are ignored.
stretch	real	Stretch ratio of element diagonals for element deletion
theory	integer	Shell theory to use
tshell	integer	Thermal shell option
wmode	real	W-mode amplitude for element deletion (deg)
wrpang	real	Shell warpage angle (deg)

*CONTROL_SOLID

Properties for *CONTROL_SOLID

Name	**Type**	**Description**
esort	integer	Automatic sort of tetra & penta flag
exists	logical	true if control card exists
fmatrix	integer	calculation method for deformation gradient
icoh	integer	global flag for cohesive element deletion
include	integer	The Include file number that the control card is in.
niptets	integer	#intg points for quadratic tets
pm1	integer	10 noded tetrahedral solid node ID 1
pm10	integer	10 noded tetrahedral solid node ID 10
pm2	integer	10 noded tetrahedral solid node ID 2
pm3	integer	10 noded tetrahedral solid node ID 3
pm4	integer	10 noded tetrahedral solid node ID 4
pm5	integer	10 noded tetrahedral solid node ID 5
pm6	integer	10 noded tetrahedral solid node ID 6
pm7	integer	10 noded tetrahedral solid node ID 7
pm8	integer	10 noded tetrahedral solid node ID 8

pm9	integer	10 noded tetrahedral solid node ID 9
psfail	integer	Optional part set id
swlocl	integer	output flag for stresses in solid spotwelds
t10jtol	real	tolerance for jacobian in 4-point 10-noded quadratic tetrahedra
tet13k	integer	global flag for cohesive element deletion
tet13v	integer	Choice of type 13 solid implementation

*CONTROL_SOLUTION

Properties for *CONTROL_SOLUTION

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
isnan	integer	Flag to check for a NaN in force and moment arrays
lcacc	integer	Flag to truncate curves: 0 = no truncation; otherwise = truncate
lcint	integer	Number of points in load curve discretization
ncdcf	integer	Cycle number at which to evaluate DEFINE_CURVE_FUNCTION
nlq	integer	Vector length
soln	integer	Solution type flag

*CONTROL_SPH

Properties for *CONTROL_SPH

Name	Type	Description
boxid	integer	Box limiting application
cont	integer	Particle approx method
deriv	integer	Time integration type
dt	real	Death time
exists	logical	true if control card exists
form	integer	particle theory
iavis	integer	artificial viscosity formulation
icont	integer	contact option
idim	integer	Space system flag
ierod	integer	erosion option
include	integer	The Include file number that the control card is in.
ini	integer	bucket or global smoothing
ishift	integer	apply shifting algorithm
ishow	integer	display option

istab	integer	stabilisation type
isymp	integer	percentage of sph
ithk	integer	contact thickness option
maxv	real	max velocity
memory	integer	memory alloc
ncbs	integer	Number of cycles between particle sorting
nmneigh	integer	memory alloc
ql	real	quasi-linear coefficient
sphsort	integer	sort and move SPH
start	real	start time

*CONTROL_SPH_INCOMPRESSIBLE

Properties for *CONTROL_SPH_INCOMPRESSIBLE

Name	Type	Description
exists	logical	true if control card exists
ibndp	integer	Pressure treatment of boundary particles
include	integer	The Include file number that the control card is in.
rol	real	Deactivation criteria
tavg	real	Tolerance criteria for average relative density
tmax	real	Tolerance criteria for maximum relative density

*CONTROL_SPOTWELD_BEAM

Properties for *CONTROL_SPOTWELD_BEAM

Name	Type	Description
bmsid	integer	beam set for convert to hex assembly
exists	logical	true if control card exists
id_off	integer	part id offset
include	integer	The Include file number that the control card is in.
lcs	integer	Loadcurve: shear response vs. shell size
lct	integer	Loadcurve: tension response vs. shell size
prtflg	integer	Flag to print data for spotwelds
rpbhx	integer	Replace each beam with a cluster of RPBHX solids
t_ors	integer	Table ID for scaling shear response
t_ort	integer	Table for scaling response

*CONTROL_STAGED_CONSTRUCTION

Properties for *CONTROL_STAGED_CONSTRUCTION

Name	Type	Description
accel	real	gravity
dordel	integer	Dormant part treatment in d3plot file
exists	logical	true if control card exists
fact	real	default stiffness/gravity factor
include	integer	The Include file number that the control card is in.
itime	integer	Treatment of "Real time" on *DEFIN_CONSTRUCTION_STAGES
nopdel	integer	Treatment of pressure loads on deleted elements
stge	integer	end stage
stgs	integer	start stage
stref	integer	ref stage
tstart	real	start time

*CONTROL_START

Properties for *CONTROL_START

Name	Type	Description
begtim	real	start time
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.

*CONTROL_STEADY_STATE_ROLLING

Properties for *CONTROL_STEADY_STATE_ROLLING

Name	Type	Description
exists	logical	true if control card exists
imass	integer	Inertia switching flag
include	integer	The Include file number that the control card is in.
ivel	integer	Velocity switching flag
lcdmu	integer	Loadcurve for scaling friction forces
lcdmur	integer	Loadcurve for scaling friction forces during dynamic relaxation
scl_k	integer	Scale factor for friction stiffness

*CONTROL_STRUCTURED

Properties for *CONTROL_STRUCTURED

Name	Type	Description

exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
term	integer	_TERM flag

*CONTROL_TERMINATION

Properties for *CONTROL_TERMINATION

Name	Type	Description
dtmin	real	Scale factor on initial dt size for termination
endcyc	integer	Termination cycle #
endeng	real	%age change in energy for termination
endmas	real	%age change in mass for termination
endtim	real	Termination time
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
nosol	integer	flag for non-solution run

*CONTROL_THERMAL_EIGENVALUE

Properties for *CONTROL_THERMAL_EIGENVALUE

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
neig	integer	Number of eigen values to compute.

*CONTROL_THERMAL_FORMING

Properties for *CONTROL_THERMAL_FORMING

Name	Type	Description
a	integer	Load curve ID for the a coefficient used in the formula
algo	integer	Contact algorithm type
b	integer	Load curve ID for the b coefficient used in the formula
bc_flg	integer	Thermal boundary condition flag
c	integer	Load curve ID for the c coefficient used in the formula
d	integer	Explicit accuracy parameter
exists	logical	true if control card exists
formula	integer	Formula that defines the contact heat conductance as a function of temperature and pressure
frad	real	Radiation factor between the contact surfaces

ftoslv	real	Fraction of sliding friction energy partitioned to the slave surface
fwork	real	Fraction of mechanical work converted into heat
h0	real	Heat transfer conductance for closed gaps
include	integer	The Include file number that the control card is in.
ithoff	integer	Flag for offsetting thermal contact surfaces for thick thermal shells
its	real	Initial thermal time step size
k	real	Thermal conductivity of fluid between the contact surfaces
lcfdt	integer	Load curve number for dynamic coefficient of friction as a function of temperature
lcfst	integer	Load curve number for static coefficient of friction as a function of temperature
lch	integer	Load curve ID for h (can be curve ID or function ID)
lmax	real	No thermal contact if gap is greater than this value
lmin	real	Minimum gap
ptype	integer	Thermal problem type
solver	integer	Thermal analysis solver type
thshel	integer	Thermal shell option
tsf	real	Thermal Speedup Factor

*CONTROL_THERMAL_NONLINEAR

Properties for *CONTROL_THERMAL_NONLINEAR

Name	Type	Description
dcp	real	Divergence control parameter
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
lumpbc	integer	lump boundary condition
nlthpr	integer	Thermal nonlinear printout level
phchpn	real	Phase change penalty parameter
refmax	integer	Max #matrix reformations per timestep
thlstl	real	Line search convergence tolerance
tol	real	Convergence tolerance for temperature

*CONTROL_THERMAL_SOLVER

Properties for *CONTROL_THERMAL_SOLVER

Name	Type	Description
abstol	real	Absolute convergence tolerance
atype	integer	Thermal analysis type
cgtol	real	Convergence tolerance for iterative solver

dtvf	real	Time interval between view factor updates
eqheat	integer	Mechanical equivelent of heat (J/Nm etc). Loadcurve if negative
exists	logical	true if control card exists
fwork	real	Fraction of mechanical heat converted into heat
gpt	integer	#gauss points in solids
include	integer	The Include file number that the control card is in.
maxitr	integer	Maximum number of iterations
msglvl	integer	Output message level
mxdmp	integer	Matrix dumping.
omega	real	Relaxation parameter
ptype	integer	Thermal problem type
reltol	real	Relative convergence tolerance
sbc	real	Stefan Boltzman constant (w/m**2/K)
solver	integer	Thermal analysis solver type
tsf	integer	Thermal speedup factor. Loadcurve if negative
varden	integer	Variable thermal density flag

*CONTROL_THERMAL_TIMESTEP

Properties for *CONTROL_THERMAL_TIMESTEP

Name	Type	Description
dtemp	integer	Max delta temp permitted before timestep decrease, of LC of dt vs time if -ve
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
its	real	Initial thermal timestep
lcts	integer	Loadcurve: timestep vs time
tip	real	Thermal time integration parameter
tmax	real	Maximum thermal timestep, or LC of tmax vs time if -ve
tmin	real	Minimum thermal timestep, or LC of tmin vs time if -ve
ts	integer	Thermal timestep control flag
tscp	real	Timestep control parameter

*CONTROL_TIMESTEP

Properties for *CONTROL_TIMESTEP

Name	Type	Description
dt2ms	real	Timestep for mass scaling
dt2msf	real	Scale factor for initial timestep size to determine min permitted time step size
dt2mslc	integer	Loadcurve: DT2MS vs time

dtinit	real	Initial timestep size
erode	integer	Erosion flag for solids & shells @ DTMIN
exists	logical	true if control card exists
ihdo	integer	Method for calculating solid element time steps
imscl	integer	Selective mass scaling. Part set if negative
include	integer	The Include file number that the control card is in.
isdo	integer	dt calc method for 4 noded shells
lctm	integer	Loadcurve: Max timestep vs time
ms1st	integer	Limit mass scaling to 1st timestep flag
rmscl	integer	flag to activate scaling of rotational inertia
tslimt	real	Min timestep for shell modulus change
tssfac	real	Scale factor for computed timestep

*CONTROL_UNITS

Properties for *CONTROL_UNITS

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
length	string	m = meter, mm = millimeter, cm = centimeter, in = inch, ft = foot
length_ scale	real	Number of meters in the length unit for the input deck
mass	string	kg = kilogram, g = gram, mg = milligram, lb = pound, slug = pound x sec2/foot, slinch = pound x sec2/inch, mtrc_ton = metric_ton
mass_ scale	real	Number of kilograms in the mass unit for the input deck
temp	string	K = Kelvin, C = Celsius, F = Fahrenheit, R = Rankine
time	string	sec = second, ms = msec/millisec, micro_s = microsec
time_scale	real	Number of seconds in the time unit for the input deck

*CONTROL_VIBRO_ACOUSTIC

Properties for *CONTROL_VIBRO_ACOUSTIC

Name	Type	Description
exists	logical	true if control card exists
include	integer	The Include file number that the control card is in.
ipanelu	integer	Number of strips in U direction
ipanelv	integer	Number of strips in V direction
nmdstr	integer	Number of modes in modal stress/strain output
restrt	integer	Restart option

vaflag	integer	Loading type
vaplot	integer	Flag for PSD broadband plots
vaprld	integer	Flag for including preload
vapsd	integer	Flag for PSD output
varms	integer	Flag for RMS output
vastrs	integer	Flag for including stress analysis

Damping class

The Damping class gives you access to damping cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Damping properties

Name	Type	Description
global	Object	*DAMPING_GLOBAL card

Properties for GLOBAL

Name	Type	Description
exists	logical	true if damping card exists
include	integer	The Include file number that the damping card is in.
lcid	integer	Curve ID specifying system damping constant
srx	real	Scale factor on global x rotational damping moments
sry	real	Scale factor on global y rotational damping moments
srz	real	Scale factor on global z rotational damping moments
stx	real	Scale factor on global x translational damping forces
sty	real	Scale factor on global y translational damping forces
stz	real	Scale factor on global z translational damping forces
valdmp	real	System damping constant

Detailed Description

The Damping class allows you to create, modify, edit and manipulate damping cards. Unlike other classes there is no constructor and there are no functions. Instead a Damping object is available as the damping property of a Model object. This object allows you to access the damping cards.
For example, to activate damping card *DAMPING_GLOBAL in model m and set valdmp to 0.001.

```
m.damping.global.exists = true;
m.damping.global.valdmp = 0.001;
```

See the properties for more details.

Database class

The Database class gives you access to database cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Database properties

Name	Type	Description
abstat	Object	*DATABASE_ABSTAT card
atdout	Object	*DATABASE_ATDOUT card
bearing	Object	*DATABASE_BEARING card
binary	Object	*DATABASE_BINARY cards
bndout	Object	*DATABASE_BNDOUT card
dcfail	Object	*DATABASE_DCFAIL card
defgeo	Object	*DATABASE_DEFGEO card
deforc	Object	*DATABASE_DEFORC card
destat	Object	*DATABASE_DESTAT card
elout	Object	*DATABASE_ELOUT card
envelope	Object	*DATABASE_ENVELOPE card
extent_binary	Object	*DATABASE_EXTENT_BINARY card
extent_binary_comp	Object	*DATABASE_EXTENT_BINARY_COMP card
extent_d3part	Object	*DATABASE_EXTENT_D3PART card
extent_intfor	Object	*DATABASE_EXTENT_INTFOR card
format	Object	*DATABASE_FORMAT card
gceout	Object	*DATABASE_GCEOUT card
glstat	Object	*DATABASE_GLSTAT card
h3out	Object	*DATABASE_H3OUT card
jntforc	Object	*DATABASE_JNTFORC card
matsum	Object	*DATABASE_MATSUM card
ncforc	Object	*DATABASE_NCFORC card
nodfor	Object	*DATABASE_NODFOR card
nodout	Object	*DATABASE_NODOUT card
pbstat	Object	*DATABASE_PBSTAT card

pllyout	Object	*DATABASE_PLLYOUT card
prtube	Object	*DATABASE_PRTUBE card
pyro	Object	*DATABASE_PYRO card
rbdout	Object	*DATABASE_RBDOUT card
rcforc	Object	*DATABASE_RCFORC card
rwforc	Object	*DATABASE_RWFORC card
sbtout	Object	*DATABASE_SBTOUT card
secforc	Object	*DATABASE_SECFORC card
sleout	Object	*DATABASE_SLEOUT card
spcforc	Object	*DATABASE_SPCFORC card
sphmassflow	Object	*DATABASE_SPHMASSFLOW card
sphout	Object	*DATABASE_SPHOUT card
swforc	Object	*DATABASE_SWFORC card
tprint	Object	*DATABASE_TPRINT card
trhist	Object	*DATABASE_TRHIST card

Properties for ABSTAT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for ATDOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for BEARING

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists

include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for BINARY

Name	Type	Description
blstfor	Object	*DATABASE_BINARY_BLSTFOR card
cpmfor	Object	*DATABASE_BINARY_CPMFOR card
d3crack	Object	*DATABASE_BINARY_D3CRACK card
d3drlf	Object	*DATABASE_BINARY_D3DRLF card
d3dump	Object	*DATABASE_BINARY_D3DUMP card
d3mean	Object	*DATABASE_BINARY_D3MEAN card
d3part	Object	*DATABASE_BINARY_D3PART card
d3plot	Object	*DATABASE_BINARY_D3PLOT card
d3prop	Object	*DATABASE_BINARY_D3PROP card
d3thdt	Object	*DATABASE_BINARY_D3THDT card
demfor	Object	*DATABASE_BINARY_DEMFOR card
fsifor	Object	*DATABASE_BINARY_FSIFOR card
fsilnk	Object	*DATABASE_BINARY_FSILNK card
intfor	Object	*DATABASE_BINARY_INTFOR card
isphfor	Object	*DATABASE_BINARY_ISPHFOR card
runrsf	Object	*DATABASE_BINARY_RUNRSF card
xtfile	Object	*DATABASE_BINARY_XTFILE card

Properties for BINARY_BLSTFOR

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_CPMFOR

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_D3CRACK

Name	Type	Description
dt	real	Time interval between outputs

exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_D3DRLF

Name	Type	Description
cycl	integer	Output interval in cycles
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_D3DUMP

Name	Type	Description
cycl	integer	Output interval in cycles
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_D3MEAN

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
iavg	integer	Averaging time interval
include	integer	The Include file number that the database binary card is in.
istats	integer	Level of statistics
tstart	real	Start time

Properties for BINARY_D3PART

Name	Type	Description
beam	integer	Beam option
bsetid	integer	Beam Set ID
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
hsetid	integer	Solid Set ID
include	integer	The Include file number that the database binary card is in.
lcdt	integer	Curve ID giving time interval between dumps
npltc	integer	Number of plot files
psetid	integer	Part Set ID
ssetid	integer	Shell Set ID
tsetid	integer	TShell Set ID

Properties for BINARY_D3PLOT

Name	Type	Description
beam	integer	Beam option
cutoff	real	Frequency cut-off C in Hz
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.
ioopt	integer	Flag for lcdt behaviour
lcdt	integer	Curve ID giving time interval between dumps
npltc	integer	Number of plot files
pset	integer	Part set ID for filtering
psetid	integer	Part Set ID
rate	real	Time interval T between filter sampling
type	integer	Flag for filtering options
window	real	Width of the window in units of time for storing single, forward filtering

Properties for BINARY_D3PROP

Name	Type	Description
exists	logical	true if database binary card exists
ifile	integer	Output data flag
imatl	integer	Output *EOS, *HOURGLASS, *MAT, *part and *SECTION data
include	integer	The Include file number that the database binary card is in.
iwall	integer	Output *RIGIDWALL data

Properties for BINARY_D3THDT

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.
lcdt	integer	Curve ID giving time interval between dumps

Properties for BINARY_DEMFOR

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_FSIFOR

Name	Type	Description
dt	real	Time interval between outputs

exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_FSILNK

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_INTFOR

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
fname	string	Filename of the database for the INTFOR data
include	integer	The Include file number that the database binary card is in.
ioopt	integer	governs how the plot state frequency is determined from curve LCDT
lcdt	integer	Curve ID giving time interval between dumps

Properties for BINARY_ISPHFOR

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BINARY_RUNRSF

Name	Type	Description
cycl	integer	Output interval in cycles
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.
nr	integer	Number of running restart files

Properties for BINARY_XTFILE

Name	Type	Description
dt	real	Time interval between outputs
exists	logical	true if database binary card exists
include	integer	The Include file number that the database binary card is in.

Properties for BNDOUT

Name	Type	Description

binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for CURVOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for DCFAIL

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for DEFGEO

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for DEFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists

include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for DESTAT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for DISBOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for ELOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval
option1	integer	extra history variables for solids
option2	integer	extra history variables for shells
option3	integer	extra history variables for thick shells
option4	integer	extra history variables for beams

Properties for ENVELOPE

Name	Type	Description
bsetid	integer	Output for beam elements. +n is output for elements in beam set n, 0 no beam, -1 all elements
exists	logical	true if database card exists

include	integer	The Include file number that the database card is in.
output	integer	Output format. Can be 0 or 1
ssetid	integer	Output for shell elements. +n is output for elements in shell set n, 0 no shell, -1 all elements
tback	real	Time interval for backup output files during the analysis
tcheck	real	Time interval for checking whether the previous maxima/minima are exceeded

Properties for EXTENT_BINARY

Name	Type	Description
beamip	integer	#beam int points to output
cmpflg	integer	Flag to output composite material stress in local csys
cubsld	integer	Output flag for quadratic solid types
dcomp	integer	Data compression flag
dtdt	integer	output of nodal temp
engflg	integer	Flag to in/exclude shell energy & thickness
epsflg	integer	Flag to in/exclude shell strains
exists	logical	true if database card exists
hydro	integer	adds extra history variables
ialemat	integer	output ale materials
ieverp	integer	Every D3PLOT file to separate database flag
include	integer	The Include file number that the database card is in.
intout	string	output of intg pt data
maxint	integer	#integration points for shell output
msscl	integer	output nodal mass scaling data
n3thdt	integer	Output for material energies to D3THDT file
neipb	integer	Output of loop-stresses to d3plot
neiph	integer	#extra values for solids
neips	integer	#extra values for shells
nintsld	integer	number of solid integration pts
nodout	string	output of connectivity nodes
pkp_sen	integer	Flag to output peak pressure and surface energy for each contact interface
quadsld	integer	Output flag for cubic solid types
resplt	integer	Output of residual forces
rltflg	integer	Flag to in/exclude shell force/moment resultants
sclp	real	Scaling parameter used in the computation of the peak pressure
shge	integer	Shell hourglass energy output flag
sigflg	integer	Flag to in/exclude shell stress tensors
strflg	integer	Strain tensor output flag
stssz	integer	Output shell element dt flag
therm	integer	Output of thermal data to d3plot

Properties for EXTENT_BINARY_COMP

Name	Type	Description
exists	logical	true if database card exists
iacc	string	output of accleration data
iglb	string	output of global data
include	integer	The Include file number that the database card is in.
ised	string	output of strain energy density data
istra	string	output of strain data
istrs	string	output of stress data
ivel	string	output of velocity data
ixyz	string	output of geometry data

Properties for EXTENT_D3PART

Name	Type	Description
engflg	integer	Flag to in/exclude shell energy & thickness
epsflg	integer	Flag to in/exclude shell strains
exists	logical	true if database card exists
ieverp	integer	Every D3PLOT file to separate database flag
include	integer	The Include file number that the database card is in.
maxint	integer	#integration points for shell output
neiph	integer	#extra values for solids
neips	integer	#extra values for shells
nintsld	integer	number of solid integration pts
rltflg	integer	Flag to in/exclude shell force/moment resultants
shge	integer	Shell hourglass energy output flag
sigflg	integer	Flag to in/exclude shell stress tensors
strflg	integer	Strain tensor output flag
stssz	integer	Output shell element dt flag

Properties for EXTENT_INTFOR

Name	Type	Description
exists	logical	true if database card exists
ieverf	integer	Every INTFOR database to separate file flag
include	integer	The Include file number that the database card is in.
neng	integer	Output contact energy density for mortar contact and SOFT = 2 contact
nfail	integer	Display deleted contact segments flag
nforc	integer	Output forces
ngapc	integer	Output contact gaps

nglbv	integer	Output global variables
nhuf	integer	Number of user friction history variables to output from user defined friction routines
npen	integer	Output penetration information for mortar contact
npresu	integer	Output pressures
nshear	integer	Output shear stresses
ntied	integer	Output tied segments for Mortar contact
nvelo	integer	Output nodal velocity
nwear	integer	Output contact wear data mode
nwrk	integer	Output (total) sliding interface energy density for mortar contact
nwusr	integer	Number of user wear history variables

Properties for FORMAT

Name	Type	Description
exists	logical	true if database card exists
ibinary	integer	Word size for binary output files
iform	integer	Output format for D3PLOT and D3THDT files
include	integer	The Include file number that the database card is in.

Properties for GCEOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for GLSTAT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval
mass_properties	integer	Flag to include mass and inertia properties

Properties for H3OUT

Name	Type	Description

binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for JNTFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for MATSUM

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for NCFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for NODFOR

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists

include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for NODOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval
option1	real	High frequency output interval
option2	integer	Flag for binary file for high frequency output

Properties for PBSTAT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for PLLYOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for PRTUBE

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.

ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for PYRO

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for RBDOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for RCFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for RWFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SBTOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SECFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SLEOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SPCFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SPHMASSFLOW

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output

exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SPHOUT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for SWFORC

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for TPRINT

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve
lcur	integer	Curve ID specifying time interval

Properties for TRHIST

Name	Type	Description
binary	integer	Flag for binary file
dt	real	Time interval between output
exists	logical	true if database card exists
include	integer	The Include file number that the database card is in.
ioopt	integer	Flag for behaviour of load curve

lcur	integer	Curve ID specifying time interval

Detailed Description

The Database class allows you to create, modify, edit and manipulate database cards. Unlike other classes there is no constructor and there are no functions. Instead a Database object is available as the database property of a Model object. This object allows you to access all of the database cards.

For example, to activate database card *DATABASE_SWFORC in model m and set dt to 0.001.

```
m.database.swforc.exists = true;
m.database.swforc.dt = 0.001;
```

See the properties for more details.

CrossSection class

The CrossSection class gives you access to database cross section cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- ElemCut(Shell label*[integer]*)
- Error(message*[string]*, details (optional)*[string]*)
- FlagCut(Flag*[Flag]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- PartCut(Part label*[integer]*, Flag (optional)*[Flag]*)
- Previous()
- Properties()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()

- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

CrossSection constants

Name	Description
CrossSection.PLANE	PLANE is *DATABASE_CROSS_SECTION_PLANE.
CrossSection.SET	SET is *DATABASE_CROSS_SECTION_SET.

CrossSection properties

Name	Type	Description
bsid	integer	Beam set number.
csid	integer	Database cross section number (identical to label).
dsid	integer	Discrete set number.
exists	logical	true if database cross section exists, false if referred to but not defined. (read only)
heading	string	Database cross section heading.
hsid	integer	Solid set number.
id	integer	Rigid part or accelerometer or coordinate system number.
idset	logical	true if _ID option is set, false if not
include	integer	The Include file number that the database cross section is in.
itype	integer	Flag for local system type.
label	integer	Database cross section number.
lenl	real	Length of L edge.
lenm	real	Length of M edge.
model	integer	The Model number that the cross section is in.
nsid	integer	Node set number.
option	constant	The Database CrossSection option. Can be: • CrossSection.PLANE or • CrossSection.SET
psid	integer	Part set number.
radius	real	Radius.
ssid	integer	Shell set number.
tsid	integer	Thick shell set number.
xch	real	Head X coord of N normal vector.
xct	real	Tail X coord of N normal vector.
xhev	real	Head X coord of L edge vector.
ych	real	Head Y coord of N normal vector.
yct	real	Tail Y coord of N normal vector.
yhev	real	Head Y coord of L edge vector.
zch	real	Head Z coord of N normal vector.

zct	real	Tail Z coord of N normal vector.
zhev	real	Head Z coord of L edge vector.

Detailed Description

The CrossSection class allows you to create, modify, edit and manipulate database cross section cards. See the documentation below for more details.

Constructor

new CrossSection(Model*[Model]*, option*[constant]*, nsid*[integer]*, hsid*[integer]*, bsid*[integer]*, ssid*[integer]*, tsid*[integer]*, dsid*[integer]*, id (optional)*[integer]*, itype (optional)*[integer]*, csid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new CrossSection object for *DATABASE_CROSS_SECTION_SET.

Arguments

Name	Type	Description
Model	Model	Model that database cross section will be created in
option	constant	Database cross section type. Must be CrossSection.SET
nsid	integer	Node set number.
hsid	integer	Solid set number.
bsid	integer	Beam set number.
ssid	integer	Shell set number.
tsid	integer	Thick shell set number.
dsid	integer	Discrete set number.
id (optional)	integer	Rigid part or accelerometer or coordinate system number.
itype (optional)	integer	Flag for local system type.
csid (optional)	integer	Database cross_section number.
heading (optional)	string	Database cross_section title.

Return type

CrossSection object

Example

To create a new Database cross section 500 called "test cross_section" of type _SET in model m with nsid, hsid, bsid, ssid, tsid, dsid, id, itype set to 11, 12, 13, 14, 15, 16, 17, 2 respectively:

```
var c = new CrossSection(m, CrossSection.SET, 11, 12, 13, 14, 15, 16, 17, 2,
500, "test cross_section");
```

new CrossSection(Model*[Model]*, option*[constant]*, psid*[integer]*, xct*[real]*, yct*[real]*, zct*[real]*, xch*[real]*, ych*[real]*, zch*[real]*, xhev*[real]*, yhev*[real]*, zhev*[real]*, lenl (optional)*[real]*, lenm (optional)*[real]*, id (optional)*[integer]*, itype (optional)*[integer]*, csid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new CrossSection object for *DATABASE_CROSS_SECTION_PLANE.

Arguments

Name	Type	Description
Model	Model	Model that database cross section will be created in
option	constant	Database cross section type. Must be CrossSection.PLANE
psid	integer	Part set number.
xct	real	X coordinate of tail of normal vector.
yct	real	Y coordinate of tail of normal vector.
zct	real	Z coordinate of tail of normal vector.
xch	real	X coordinate of head of normal vector.
ych	real	Y coordinate of head of normal vector.
zch	real	Z coordinate of head of normal vector.
xhev	real	X coordinate of head of edge vector.
yhev	real	Y coordinate of head of edge vector.
zhev	real	Z coordinate of head of edge vector.
lenl (optional)	real	Length in l direction.
lenm (optional)	real	Length in m direction.
id (optional)	integer	Rigid part or accelerometer or coordinate system number.
itype (optional)	integer	Flag for local system type.
csid (optional)	integer	Database cross_section number.
heading (optional)	string	Database cross_section title.

Return type

CrossSection object

Example

To create a new Database cross section 500 called "test cross_section" of type _PLANE in model m with part set ID 100, normal tail (10, 20, 30), normal head (20, 20, 30), head of edge vector (10, 30, 30) and edge lengths 50 and 100:

```
var c = new CrossSection(m, CrossSection.PLANE, 100, 10, 20, 30, 20, 20, 30, 10, 30, 30, 50, 100, 0, 0, 500, "test cross_section");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a cross section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the cross section

Return type

No return value

Example

To associate comment c to the cross section c:

```
c.AssociateComment(c);
```

Blank()

Description

Blanks the cross section

Arguments

No arguments

Return type

No return value

Example

To blank cross section c:

```
c.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the cross sections in model m:

```
CrossSection.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged cross sections will be blanked in
flag	Flag	Flag set on the cross sections that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the cross sections in model m flagged with f:

```
CrossSection.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the cross section is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if cross section c is blanked:

```
if (c.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse cross section c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the cross section.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the cross section

Return type

No return value

Example

To clear flag f for cross section c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the cross section.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

CrossSection object

Example

To copy cross section c into cross section z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a cross_section.

Arguments

Name	Type	Description
Model	Model	Model that the cross_section will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

CrossSection object (or null if not made)

Example

To start creating a cross section in model m:

```
var c = CrossSection.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a cross section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the cross section

Return type

No return value

Example

To detach comment c from the cross section c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit cross section c:

```
c.Edit();
```

ElemCut(Shell label*[integer]*)

Description

Returns coordinates of the intersections between a shell and a database cross section. Note, ElemCut on the Shell class may be quicker

Arguments

Name	Type	Description
Shell label	integer	The label of the shell.

Return type

An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line, or NULL if it does not cut. Note this function does not check that the shell is in the cross section definition (part set)

Example

To get the cut line coordinates between database cross section x and shell 300:

```
var data = x.ElemCut(300)
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for cross section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for cross section c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first cross section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first cross section in

Return type

CrossSection object (or null if there are no cross sections in the model).

Example

To get the first cross section in model m:

```
var c = CrossSection.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free cross section label in the model. Also see CrossSection.LastFreeLabel(), CrossSection.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free cross section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

CrossSection label.

Example

To get the first free cross section label in model m:

```
var label = CrossSection.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the cross sections in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections will be flagged in
flag	Flag	Flag to set on the cross sections

Return type

No return value

Example

To flag all of the cross sections with flag f in model m:

```
CrossSection.FlagAll(m, f);
```

FlagCut(Flag*[Flag]*)

Description

Flags every element (solid,shell,tshell,beam) cut by the cross section. Note this function does not check that the element is in the cross section definition (part set)

Arguments

Name	Type	Description
Flag	Flag	Flag bit.

Return type

Boolean.

Example

To find elements cut by database cross section x:

```
x.FlagCut(flag)
```

Flagged(flag*[Flag]*)

Description

Checks if the cross section is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the cross section

Return type

true if flagged, false if not.

Example

To check if cross section c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each cross section in the model.
Note that ForEach has been designed to make looping over cross sections as fast as possible and so has some limitations.
Firstly, a single temporary CrossSection object is created and on each function call it is updated with the current cross section data. This means that you should not try to store the CrossSection object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new cross sections inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections are in
func	function	Function to call for each cross section
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the cross sections in model m:

```
CrossSection.ForEach(m, test);
function test(c)
{
// c is CrossSection object
}
```

To call function test for all of the cross sections in model m with optional object:

```
var data = { x:0, y:0 };
CrossSection.ForEach(m, test, data);
function test(c, extra)
{
// c is CrossSection object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of CrossSection objects for all of the cross sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get cross sections from

Return type

Array of CrossSection objects

Example

To make an array of CrossSection objects for all of the cross sections in model m

```
var c = CrossSection.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a cross section.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the cross section c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of CrossSection objects for all of the flagged cross sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get cross sections from
flag	Flag	Flag set on the cross sections that you want to retrieve

Return type

Array of CrossSection objects

Example

To make an array of CrossSection objects for all of the cross sections in model m flagged with f

```
var c = CrossSection.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the CrossSection object for a cross section ID.

Arguments

Name	Type	Description
Model	Model	Model to find the cross section in
number	integer	number of the cross section you want the CrossSection object for

Return type

CrossSection object (or null if cross section does not exist).

Example

To get the CrossSection object for cross section 100 in model m

```
var c = CrossSection.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a CrossSection property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the CrossSection.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	cross section property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if CrossSection property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if CrossSection property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this cross_section (*DATABASE_CROSS_SECTION). **Note that a carriage return is not added**. See also CrossSection.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for cross_section c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the cross_section. **Note that a carriage return is not added**. See also CrossSection.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for cross_section c:

```
var cards = c.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last cross section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last cross section in

Return type

CrossSection object (or null if there are no cross sections in the model).

Example

To get the last cross section in model m:

```
var c = CrossSection.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free cross section label in the model. Also see CrossSection.FirstFreeLabel(), CrossSection.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free cross section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

CrossSection label.

Example

To get the last free cross section label in model m:

```
var label = CrossSection.LastFreeLabel(m);
```

Next()

Description

Returns the next cross section in the model.

Arguments

No arguments

Return type

CrossSection object (or null if there are no more cross sections in the model).

Example

To get the cross section in model m after cross section c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) cross section label in the model. Also see CrossSection.FirstFreeLabel(), CrossSection.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free cross section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

CrossSection label.

Example

To get the next free cross section label in model m:

```
var label = CrossSection.NextFreeLabel(m);
```

PartCut(Part label*[integer]*, Flag (optional)*[Flag]*)

Description

Returns true if cross section is cutting the part, false otherwise. If option flag is active, will flag every element of the part cut by the cross section. Note this function does not check that the part is in the cross section definition (part set)

Arguments

Name	Type	Description
Part label	integer	The label of the part.
Flag (optional)	Flag	Optional Flag to flag the element which are cut by the cross section.

Return type

Boolean.

Example

To know if a database cross section x cuts part 300:

```
x.PartCut(300)
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a cross section.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only cross sections from that model can be picked. If the argument is a Flag then only cross sections that are flagged with *limit* can be selected. If omitted, or null, any cross sections from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

CrossSection object (or null if not picked)

Example

To pick a cross section from model m giving the prompt 'Pick cross section from screen':

```
var c = CrossSection.Pick('Pick cross section from screen', m);
```

Previous()

Description

Returns the previous cross section in the model.

Arguments

No arguments

Return type

CrossSection object (or null if there are no more cross sections in the model).

Example

To get the cross section in model m before cross section c:

```
var c = c.Previous();
```

Properties()

Description

Returns an object which describe various cross section properties

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
area	real	Area of material sliced by the cut section
first_yield_axial	real	First yield axial force
first_yield_mxx	real	First yield bending moment (Mxx)
first_yield_myy	real	First yield bending moment (Myy)
fully_plastic_mxx	real	Fully plastic bending moment (Mxx)
fully_plastic_myy	real	Fully plastic bending moment (Myy)
fully_plastic_xf	real	Eq force axis Xf
fully_plastic_yf	real	Fully plastic axial force
iuu	real	Iuu principal second moments (UU - major)
ivv	real	Ivv principal second moments (VV - minor)
ixx	real	Ixx component of second moment of area
ixy	real	Ixy component of second moment of area
iyy	real	Iyy component of second moment of area
origin_x	real	X component of section origin
origin_y	real	Y component of section origin
origin_z	real	Z component of section origin
theta	real	Angle between Ixx and Iuu
x_comp_axis_x	real	X component of X-axis vector
x_comp_axis_y	real	X component of Y-axis vector
x_comp_axis_z	real	X component of Z-axis vector

xc	real	X component of centroid calculated from the first moment of area
xc_global	real	X component of centre of gravity calculated in global coordinates
xe	real	X component of equal area axis
y_comp_axis_x	real	Y component of x-axis vector
y_comp_axis_y	real	Y component of Y-axis vector
y_comp_axis_z	real	Y component of Z-axis vector
yc	real	Y component of centroid calculated from the first moment of area
yc_global	real	Y component of centre of gravity calculated in global coordinates
ye	real	Y component of equal area axis
z_comp_axis_x	real	Z component of X-axis vector
z_comp_axis_y	real	Z component of Y-axis vector
z_comp_axis_z	real	Z component of Z-axis vector
zc_global	real	Z component of centre of gravity calculated in global coordinates
zxx	real	Plastic moduli Zxx
zyy	real	Plastic moduli Zyy

Example

To get the cross section properties for section c:

```
var properties = c.Properties();
    var originX  = properties.origin_x;
    var originY  = properties.origin_y;
    var Xc = properties.xc;
    var Ixx = properties.ixx;
    var Iuu = properties.iuu;
    var Theta = properties.theta;
    var plastic_Mxx = properties.fully_plastic_mxx;
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the cross sections in model m, from 1000000:

```
CrossSection.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged cross sections will be renumbered in
flag	Flag	Flag set on the cross sections that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the cross sections in model m flagged with f, from 1000000:

```
CrossSection.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select cross sections using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting cross sections
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only cross sections from that model can be selected. If the argument is a Flag then only cross sections that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any cross sections can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of cross sections selected or null if menu cancelled

Example

To select cross sections from model m, flagging those selected with flag f, giving the prompt 'Select cross sections':

```
CrossSection.Select(f, 'Select cross sections', m);
```

To select cross sections, flagging those selected with flag f but limiting selection to cross sections flagged with flag l, giving the prompt 'Select cross sections':

```
CrossSection.Select(f, 'Select cross sections', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the cross section.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the cross section

Return type

No return value

Example

To set flag f for cross section c:

```
c.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the cross section. The cross section will be sketched until you either call CrossSection.Unsketch(), CrossSection.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the cross section is sketched. If omitted redraw is true. If you want to sketch several cross sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch cross section c:

```
c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged cross sections in the model. The cross sections will be sketched until you either call CrossSection.Unsketch(), CrossSection.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged cross sections will be sketched in
flag	Flag	Flag set on the cross sections that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the cross sections are sketched. If omitted redraw is true. If you want to sketch flagged cross sections several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all cross sections flagged with flag in model m:

```
CrossSection.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing cross sections should be counted. If false or omitted referenced but undefined cross sections will also be included in the total.

Return type

number of cross sections

Example

To get the total number of cross sections in model m:

```
var total = CrossSection.Total(m);
```

Unblank()

Description

Unblanks the cross section

Arguments

No arguments

Return type

No return value

Example

To unblank cross section c:

```
c.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the cross sections in the model.

Arguments

Name	Type	Description

Model	Model	Model that all cross sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the cross sections in model m:

```
CrossSection.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged cross sections will be unblanked in
flag	Flag	Flag set on the cross sections that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the cross sections in model m flagged with f:

```
CrossSection.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all cross sections will be unset in
flag	Flag	Flag to unset on the cross sections

Return type

No return value

Example

To unset the flag f on all the cross sections in model m:

```
CrossSection.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the cross section.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the cross section is unsketched. If omitted redraw is true. If you want to unsketch several cross sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch cross section c:

```
c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all cross sections.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the cross sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all cross sections in model m:

```
CrossSection.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged cross sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all cross sections will be unsketched in
flag	Flag	Flag set on the cross sections that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the cross sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all cross sections flagged with flag in model m:

```
CrossSection.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

CrossSection object.

Example

To check if CrossSection property c.example is a parameter by using the CrossSection.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for cross section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for cross section c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this cross section.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for cross section c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the cross_section data in keyword format. Note that this contains the keyword header and the keyword cards. See also CrossSection.Keyword() and CrossSection.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for cross_section c in keyword format

```
var s = c.toString();
```

History class

The History class gives you access to database history cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, type*[constant]*, modal (optional)*[boolean]*)
- First(Model*[Model]*, type (optional)*[constant]*)
- FlagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*)
- GetAll(Model*[Model]*, type (optional)*[constant]*)
- GetFromID(Model*[Model]*, database history number*[integer]*)
- Last(Model*[Model]*, type (optional)*[constant]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*, type (optional)*[constant]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)

Member functions

- Blanked()
- ClearFlag(flag*[Flag]*)
- Edit(modal (optional)*[boolean]*)
- Flagged(flag*[Flag]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- Xrefs()
- toString()

History constants

Name	Description
History.ACOUSTIC	ACOUSTIC is *DATABASE_HISTORY_ACOUSTIC.
History.ALL_TYPES	All *DATABASE_HISTORY_ types.
History.BEAM	BEAM is *DATABASE_HISTORY_BEAM.
History.BEAM_SET	BEAM_SET is *DATABASE_HISTORY_BEAM_SET.
History.DISCRETE	DISCRETE is *DATABASE_HISTORY_DISCRETE.
History.DISCRETE_SET	DISCRETE_SET is *DATABASE_HISTORY_DISCRETE_SET.
History.NODE	NODE is *DATABASE_HISTORY_NODE.

History.NODE_SET	NODE_SET is *DATABASE_HISTORY_NODE_SET.
History.SEATBELT	SEATBELT is *DATABASE_HISTORY_SEATBELT.
History.SHELL	SHELL is *DATABASE_HISTORY_SHELL.
History.SHELL_SET	SHELL_SET is *DATABASE_HISTORY_SHELL_SET.
History.SOLID	SOLID is *DATABASE_HISTORY_SOLID.
History.SOLID_SET	SOLID_SET is *DATABASE_HISTORY_SOLID_SET.
History.SPH	SPH is *DATABASE_HISTORY_SPH.
History.SPH_SET	SPH_SET is *DATABASE_HISTORY_SPH_SET.
History.TSHELL	TSHELL is *DATABASE_HISTORY_TSHELL.
History.TSHELL_SET	TSHELL_SET is *DATABASE_HISTORY_TSHELL_SET.

History properties

Name	Type	Description
cid	integer	Coordinate system ID for _LOCAL
exists	logical	true if database history exists, false if referred to but not defined. (read only)
heading	string	Optional heading
hfo	integer	High frequency flag for _LOCAL
id	integer	ID of the item
include	integer	The Include file number that the database history is in.
local	logical	Turns _LOCAL on or off
model	integer	The Model number that the database history is in.
ref	integer	Output reference for _LOCAL
type (read only)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET.

Detailed Description

The History class allows you to create, modify, edit and manipulate database history cards. See the documentation below for more details.

Constructor

new History(Model*[Model]*, type*[constant]*, id*[integer]*, heading (optional)*[string]*)

Description

Create a new History object.

Arguments

Name	Type	Description

Model	Model	Model that database history will be created in
type	constant	Entity type
id	integer	ID of the item
heading (optional)	string	Optional heading

Return type

History object

Example

To create a new Database history on NODE 500 called "test history":

```
var c = new History(m, History.NODE, 500, "test history");
```

Details of functions

BlankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the database histories in the model.

Arguments

Name	Type	Description
Model	Model	Model that all database histories will be blanked in
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the database histories in model m:

```
History.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged database histories in the model.

Arguments

Name	Type	Description

Model	Model	Model that all the flagged database histories will be blanked in
flag	Flag	Flag set on the database histories that you want to blank
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the database histories in model m flagged with f:

```
History.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the database history is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if database history c is blanked:

```
if (c.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the database history.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the database history

Return type

No return value

Example

To clear flag f for database history c:

```
c.ClearFlag(f);
```

Create(Model*[Model]*, type*[constant]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a database history.

Arguments

Name	Type	Description
Model	Model	Model that the database history will be created in
type	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET.
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

History object (or null if not made)

Example

To start creating a history in model m:

```
var c = History.Create(m);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel to edit the database history.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

No return value

Example

To edit database history c:

```
c.Edit();
```

First(Model*[Model]*, type (optional)*[constant]*) [static]

Description

Returns the first database history in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first database history in
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

History object (or null if there are no database histories in the model).

Example

To get the first database history in model m:

```
var history = History.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*) [static]

Description

Flags all of the database histories in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all database histories will be flagged in
flag	Flag	Flag to set on the database histories
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

No return value

Example

To flag all of the database histories with flag f in model m:

```
History.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the database history is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the database history

Return type

true if flagged, false if not.

Example

To check if database history c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

GetAll(Model*[Model]*, type (optional)*[constant]*) [static]

Description

Returns an array of History objects for all of the database histories in a models in Primer

Arguments

Name	Type	Description
Model	Model	Model to get database histories from
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

Array of History objects

Example

To make an array of History objects for all of the database histories in model m

```
var database history = History.GetAll(m);
```

GetFromID(Model*[Model]*, database history number*[integer]*) [static]

Description

Returns the History object for a database history ID.

Arguments

Name	Type	Description
Model	Model	Model to find the database history in
database history number	integer	number of the database history you want the History object for

Return type

History object (or null if database history does not exist).

Example

To get the History object for database history 100 in model m

```
var database history = History.GetFromID(m, 100);
```

Keyword()

Description

Returns the keyword for this database history (*DATABASE_HISTORY). **Note that a carriage return is not added**. See also History.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for database history c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the database history. **Note that a carriage return is not added**. See also History.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for database history c:

```
var cards = c.KeywordCards();
```

Last(Model*[Model]*, type (optional)*[constant]*) [static]

Description

Returns the last database history in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last database history in
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

History object (or null if there are no database histories in the model).

Example

To get the last database history in model m:

```
var database history = History.Last(m);
```

Next()

Description

Returns the next database history in the model.

Arguments

No arguments

Return type

History object (or null if there are no more database histories in the model).

Example

To get the database history in model m after database history c:

```
var database history = c.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to pick a database history.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only database histories from that model can be picked. If the argument is a Flag then only database histories that are flagged with *limit* can be selected. If omitted, or null, any database histories from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.

Return type

History object (or null if not picked)

Example

To pick a database history from model m giving the prompt ’Pick database history from screen’:

```
var database history = History.Pick(’Pick database history from screen’, m);
```

Previous()

Description

Returns the previous database history in the model.

Arguments

No arguments

Return type

History object (or null if there are no more database histories in the model).

Example

To get the database history in model m before this one:

```
var history = history.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select database histories using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting database histories
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only database histories from that model can be selected. If the argument is a Flag then only database histories that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any database histories from any model can be selected.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of items selected or null if menu cancelled

Example

To select database histories from model m, flagging those selected which flag f, giving the prompt 'Select database histories':

```
History.Select(f, 'Select database histories', m);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the database history.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the database history

Return type

No return value

Example

To set flag f for database history c:

```
c.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the database history. The database history will be sketched until you either call History.Unsketch(), History.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the database history is sketched. If omitted redraw is true. If you want to sketch several database histories and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch database history c:

```
c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged database histories in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged database histories will be sketched in
flag	Flag	Flag set on the database histories that you want to sketch
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is true. If you want to do several (un)sketches and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To sketch all of the database histories of type SHELL_SET in model m flagged with f:

```
History.SketchFlagged(m, f, History.SHELL_SET);
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*, type (optional)*[constant]*) [static]

Description

Unblanks all of the database histories in the model.

Arguments

Name	Type	Description
Model	Model	Model that all database histories will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

No return value

Example

To unblank all of the database histories in model m:

```
History.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged database histories in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged database histories will be unblanked in
flag	Flag	Flag set on the database histories that you want to unblank
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the database histories in model m flagged with f:

```
History.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*) [static]

Description

Unsets a defined flag on all of the database histories in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all database histories will be unset in
flag	Flag	Flag to unset on the database histories
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.

Return type

No return value

Example

To unset the flag f on all the database histories in model m:

```
History.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the database history.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the database history is unsketched. If omitted redraw is true. If you want to unsketch several database histories and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch database history c:

```
c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all database histories.

Arguments

Name	Type	Description
Model	Model	Model that all database histories will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the database histories are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all database histories in model m:

```
History.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged database histories.

Arguments

Name	Type	Description
Model	Model	Model that all database histories will be unblanked in
flag	Flag	Flag set on the database histories that you want to sketch
type (optional)	constant	The database history type. Can be History.ACOUSTIC or History.BEAM or History.BEAM_SET or History.DISCRETE or History.DISCRETE_SET or History.NODE or History.NODE_SET or History.SEATBELT or History.SHELL or History.SHELL_SET or History.SOLID or History.SOLID_SET or History.SPH or History.SPH_SET or History.TSHELL or History.TSHELL_SET or History.ALL_TYPES. If omitted, applied to all database history types.
redraw (optional)	boolean	If model should be redrawn or not after the database histories are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all of the database histories in model m flagged with f:

```
History.UnsketchFlagged(m, f);
```

Xrefs()

Description

Returns the cross references for this database history.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for this database history:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the database history data in keyword format. Note that this contains the keyword header and the keyword cards. See also History.Keyword() and History.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for database history c in keyword format

```
var s = c.toString();
```

NodalForceGroup (Nfgr) class

The NodalForceGroup class gives you access to database nodal force group cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

NodalForceGroup properties

Name	Type	Description
cid	integer	Coordinate System ID.

exists	logical	true if Nodal Force Group exists, false if referred to but not defined. (read only)
id	integer	Database Nodal Force Group number (identical to label).
include	integer	The Include file number that the Nodal Force Group is in.
label	integer	Database Nodal Force Group number.
model	integer	The Model number that the nodal force group is in.
nsid	integer	Set Node Set ID.

Detailed Description

The NodalForceGroup class allows you to create, modify, edit and manipulate nodal force group cards. See the documentation below for more details.

For convenience "Nfgr" can also be used as the class name instead of "NodalForceGroup".

Constructor

new NodalForceGroup(Model*[Model]*, nsid*[integer]*, cid (optional)*[integer]*)

Description

Create a new NodalForceGroup object.

Arguments

Name	Type	Description
Model	Model	Model that nodal force group will be created in
nsid	integer	Set Node Set ID.
cid (optional)	integer	Coordinate System ID.

Return type

NodalForceGroup object

Example

To create a new nodal force group in model m with nsid 100:

```
var nfg = new NodalForceGroup(m, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a nodal force group.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the nodal force group

Return type

No return value

Example

To associate comment c to the nodal force group nfg:

```
nfg.AssociateComment(c);
```

Blank()

Description

Blanks the nodal force group

Arguments

No arguments

Return type

No return value

Example

To blank nodal force group nfg:

```
nfg.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodal force groups in model m:

```
NodalForceGroup.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodal force groups will be blanked in

flag	Flag	Flag set on the nodal force groups that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodal force groups in model m flagged with f:

```
NodalForceGroup.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the nodal force group is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if nodal force group nfg is blanked:

```
if (nfg.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the nodal force group.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the nodal force group

Return type

No return value

Example

To clear flag f for nodal force group nfg:

```
nfg.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the nodal force group.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

NodalForceGroup object

Example

To copy nodal force group nfg into nodal force group z:

```
var z = nfg.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a nodal force group.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the nodal force group

Return type

No return value

Example

To detach comment c from the nodal force group nfg:

```
nfg.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for nodal force group. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for nodal force group nfg:

```
nfg.Error("My custom error");
```

First(Model[*Model*]) [static]

Description

Returns the first nodal force group in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first nodal force group in

Return type

NodalForceGroup object (or null if there are no nodal force groups in the model).

Example

To get the first nodal force group in model m:

```
var nfg = NodalForceGroup.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the nodal force groups in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups will be flagged in
flag	Flag	Flag to set on the nodal force groups

Return type

No return value

Example

To flag all of the nodal force groups with flag f in model m:

```
NodalForceGroup.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the nodal force group is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the nodal force group

Return type

true if flagged, false if not.

Example

To check if nodal force group nfg has flag f set on it:

```
if (nfg.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each nodal force group in the model.
Note that ForEach has been designed to make looping over nodal force groups as fast as possible and so has some limitations.
Firstly, a single temporary NodalForceGroup object is created and on each function call it is updated with the current nodal force group data. This means that you should not try to store the NodalForceGroup object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new nodal force groups inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups are in
func	function	Function to call for each nodal force group
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the nodal force groups in model m:

```
NodalForceGroup.ForEach(m, test);
function test(nfg)
{
// nfg is NodalForceGroup object
}
```

To call function test for all of the nodal force groups in model m with optional object:

```
var data = { x:0, y:0 };
NodalForceGroup.ForEach(m, test, data);
function test(nfg, extra)
{
// nfg is NodalForceGroup object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of NodalForceGroup objects for all of the nodal force groups in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodal force groups from

Return type

Array of NodalForceGroup objects

Example

To make an array of NodalForceGroup objects for all of the nodal force groups in model m

```
var nfg = NodalForceGroup.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a nodal force group.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the nodal force group nfg:

```
var comm_array = nfg.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of NodalForceGroup objects for all of the flagged nodal force groups in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodal force groups from
flag	Flag	Flag set on the nodal force groups that you want to retrieve

Return type

Array of NodalForceGroup objects

Example

To make an array of NodalForceGroup objects for all of the nodal force groups in model m flagged with f

```
var nfg = NodalForceGroup.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the NodalForceGroup object for a nodal force group ID.

Arguments

Name	Type	Description

Model	Model	Model to find the nodal force group in
number	integer	number of the nodal force group you want the NodalForceGroup object for

Return type

NodalForceGroup object (or null if nodal force group does not exist).

Example

To get the NodalForceGroup object for nodal force group 100 in model m

```
var nfg = NodalForceGroup.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a NodalForceGroup property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the NodalForceGroup.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	nodal force group property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if NodalForceGroup property nfg.example is a parameter:

```
Options.property_parameter_names = true;
if (nfg.GetParameter(nfg.example) ) do_something...
Options.property_parameter_names = false;
```

To check if NodalForceGroup property nfg.example is a parameter by using the GetParameter method:

```
if (nfg.ViewParameters().GetParameter(nfg.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this nodal force group. **Note that a carriage return is not added**. See also NodalForceGroup.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for nodal force group nfg:

```
var key = nfg.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the nodal force group. **Note that a carriage return is not added**. See also NodalForceGroup.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for nodal force group nfg:

```
var cards = nfg.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last nodal force group in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last nodal force group in

Return type

NodalForceGroup object (or null if there are no nodal force groups in the model).

Example

To get the last nodal force group in model m:

```
var nfg = NodalForceGroup.Last(m);
```

Next()

Description

Returns the next nodal force group in the model.

Arguments

No arguments

Return type

NodalForceGroup object (or null if there are no more nodal force groups in the model).

Example

To get the nodal force group in model m after nodal force group nfg:

```
var nfg = nfg.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a nodal force group.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodal force groups from that model can be picked. If the argument is a Flag then only nodal force groups that are flagged with *limit* can be selected. If omitted, or null, any nodal force groups from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

NodalForceGroup object (or null if not picked)

Example

To pick a nodal force group from model m giving the prompt 'Pick nodal force group from screen':

```
var nfg = NodalForceGroup.Pick('Pick nodal force group from screen', m);
```

Previous()

Description

Returns the previous nodal force group in the model.

Arguments

No arguments

Return type

NodalForceGroup object (or null if there are no more nodal force groups in the model).

Example

To get the nodal force group in model m before nodal force group nfg:

```
var nfg = nfg.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select nodal force groups using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting nodal force groups
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodal force groups from that model can be selected. If the argument is a Flag then only nodal force groups that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any nodal force groups can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of nodal force groups selected or null if menu cancelled

Example

To select nodal force groups from model m, flagging those selected with flag f, giving the prompt 'Select nodal force groups':

```
NodalForceGroup.Select(f, 'Select nodal force groups', m);
```

To select nodal force groups, flagging those selected with flag f but limiting selection to nodal force groups flagged with flag l, giving the prompt 'Select nodal force groups':

```
NodalForceGroup.Select(f, 'Select nodal force groups', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the nodal force group.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the nodal force group

Return type

No return value

Example

To set flag f for nodal force group nfg:

```
nfg.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the nodal force group. The nodal force group will be sketched until you either call NodalForceGroup.Unsketch(), NodalForceGroup.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the nodal force group is sketched. If omitted redraw is true. If you want to sketch several nodal force groups and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch nodal force group nfg:

```
nfg.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged nodal force groups in the model. The nodal force groups will be sketched until you either call NodalForceGroup.Unsketch(), NodalForceGroup.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodal force groups will be sketched in
flag	Flag	Flag set on the nodal force groups that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the nodal force groups are sketched. If omitted redraw is true. If you want to sketch flagged nodal force groups several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all nodal force groups flagged with flag in model m:

```
NodalForceGroup.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing nodal force groups should be counted. If false or omitted referenced but undefined nodal force groups will also be included in the total.

Return type

number of nodal force groups

Example

To get the total number of nodal force groups in model m:

```
var total = NodalForceGroup.Total(m);
```

Unblank()

Description

Unblanks the nodal force group

Arguments

No arguments

Return type

No return value

Example

To unblank nodal force group nfg:

```
nfg.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodal force groups in model m:

```
NodalForceGroup.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged nodal force groups will be unblanked in
flag	Flag	Flag set on the nodal force groups that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodal force groups in model m flagged with f:

```
NodalForceGroup.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all nodal force groups will be unset in
flag	Flag	Flag to unset on the nodal force groups

Return type

No return value

Example

To unset the flag f on all the nodal force groups in model m:

```
NodalForceGroup.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the nodal force group.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the nodal force group is unsketched. If omitted redraw is true. If you want to unsketch several nodal force groups and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch nodal force group nfg:

```
nfg.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all nodal force groups.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the nodal force groups are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodal force groups in model m:

```
NodalForceGroup.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged nodal force groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodal force groups will be unsketched in
flag	Flag	Flag set on the nodal force groups that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the nodal force groups are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodal force groups flagged with flag in model m:

```
NodalForceGroup.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

NodalForceGroup object.

Example

To check if NodalForceGroup property nfg.example is a parameter by using the NodalForceGroup.GetParameter() method:

```
if (nfg.ViewParameters().GetParameter(nfg.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for nodal force group. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for nodal force group nfg:

```
nfg.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this nodal force group.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for nodal force group nfg:

```
var xrefs = nfg.Xrefs();
```

---

toString()

Description

Creates a string containing the nodal force group data in keyword format. Note that this contains the keyword header and the keyword cards. See also NodalForceGroup.Keyword() and NodalForceGroup.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for nodal force group n in keyword format

```
var s = n.toString();
```

Box class

The Box class gives you access to define box cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- BlankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Create(Model *[Model]*, modal (optional) *[boolean]*)
- First(Model *[Model]*)
- FirstFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- FlagAll(Model *[Model]*, flag *[Flag]*)
- ForEach(Model *[Model]*, func *[function]*, extra (optional) *[any]*)
- GetAll(Model *[Model]*)
- GetFlagged(Model *[Model]*, flag *[Flag]*)
- GetFromID(Model *[Model]*, number *[integer]*)
- Last(Model *[Model]*)
- LastFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- NextFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- Pick(prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*, button text (optional) *[string]*)
- RenumberAll(Model *[Model]*, start *[integer]*)
- RenumberFlagged(Model *[Model]*, flag *[Flag]*, start *[integer]*)
- Select(flag *[Flag]*, prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*)
- SketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Total(Model *[Model]*, exists (optional) *[boolean]*)
- UnblankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnblankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- UnflagAll(Model *[Model]*, flag *[Flag]*)
- UnsketchAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnsketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)

Member functions

- AssociateComment(Comment *[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional) *[boolean]*)
- ClearFlag(flag *[Flag]*)
- Copy(range (optional) *[boolean]*)
- DetachComment(Comment *[Comment]*)
- Edit(modal (optional) *[boolean]*)
- Error(message *[string]*, details (optional) *[string]*)
- Flagged(flag *[Flag]*)
- GetComments()
- GetParameter(prop *[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag *[Flag]*)
- Sketch(redraw (optional) *[boolean]*)
- Unblank()
- Unsketch(redraw (optional) *[boolean]*)
- ViewParameters()
- Warning(message *[string]*, details (optional) *[string]*)
- Xrefs()
- toString()

Box constants

Name	Description
Box.BOX	Box is *DEFINE_BOX.
Box.BOX_ADAPTIVE	Box is *DEFINE_BOX_ADAPTIVE.
Box.BOX_COARSEN	Box is *DEFINE_BOX_COARSEN.
Box.BOX_DRAWBEAD	Box is *DEFINE_BOX_DRAWBEAD.
Box.BOX_SPH	Box is *DEFINE_BOX_SPH.

Box properties

Name	Type	Description
bid	integer	Box number. Also see the label property which is an alternative name for this.
brmax	real	Maximum mesh size in 3D tetrahedron adaptivity
brmin	real	Minimum mesh size in 3D tetrahedron adaptivity
cid	integer	Optional coordinate system ID for tubular drawbead
cx	real	X coordinate of offset vector to local origin
cy	real	Y coordinate of offset vector to local origin
cz	real	Z coordinate of offset vector to local origin
exists	logical	true if box exists, false if referred to but not defined. (read only)
heading	string	Box heading
idir	integer	Direction of tooling movement. 1: x-direction, 2: y-direction, 3: z-direction
iflag	integer	Element protection flag. 0: elements inside, 1: elements outside box cannot be coarsened.
include	integer	The Include file number that the box is in.
label	integer	Box number. Also see the bid property which is an alternative name for this.
lcid	integer	Load curve ID to describe motion value versus time
level	integer	Maximum number of refinement levels for elements contained in box
lidx	integer	Box movement in global X axis or by node. The ndid property is an alternative name for this.
lidy	integer	Box movement in global Y axis
lidz	integer	Box movement in global Z axis
local	logical	Turns _LOCAL on or off
model	integer	The Model number that the box is in.
ndid	integer	Box movement in global X axis or by node. The lidx property is an alternative name for this.
nid	integer	Referential nodal ID for vd = 2
option	constant	The box option. Can be Box.BOX, Box.BOX_ADAPTIVE, Box.BOX_COARSEN, Box.BOX_DRAWBEAD or Box.BOX_SPH.
pid_ adaptive	integer	Part ID for Box.BOX_ADAPTIVE option
pid_ drawbead	integer	Part ID of blank for Box.BOX_DRAWBEAD option
radius	real	Radius of tube centered around draw bead

sid	integer	Part set, part or node set defining the nodal points along draw bead
stype	integer	Set type for stype. 2: part set ID, 3: part ID, 4: node set ID
vd	integer	Velocity/Displacement flag. 0: velocity, 1: displacement, 2: referential node
vid	integer	Vector ID of DOF
xmn	real	Minimum X coordinate
xmx	real	Maximum X coordinate
xv	real	Local V vector X coordinate
xx	real	Local X vector X coordinate
ymn	real	Minimum Y coordinate
ymx	real	Maximum Y coordinate
yv	real	Local V vector Y coordinate
yx	real	Local X vector Y coordinate
zmn	real	Minimum Z coordinate
zmx	real	Maximum Z coordinate
zv	real	Local V vector Z coordinate
zx	real	Local X vector Z coordinate

Detailed Description

The Box class allows you to create, modify, edit and manipulate box cards. See the documentation below for more details.

Constructor

new Box(Model*[Model]*, bid*[integer]*, xmn*[real]*, xmx*[real]*, ymn*[real]*, ymx*[real]*, zmn*[real]*, zmx*[real]*, heading (optional)*[string]*)

Description

Create a new Box object.

Arguments

Name	Type	Description
Model	Model	Model that box will be created in
bid	integer	Box number
xmn	real	Minimum X coordinate
xmx	real	Maximum X coordinate
ymn	real	Minimum Y coordinate
ymx	real	Maximum Y coordinate
zmn	real	Minimum Z coordinate
zmx	real	Maximum Z coordinate
heading (optional)	string	Title for the box

Return type

Box object

Example

To create a new box in model m with label 200

```
var b = new Box(m, 200, 1.5, 2.5, 1.0, 4.5, -4.0, 3.0);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a box.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the box

Return type

No return value

Example

To associate comment c to the box b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the box

Arguments

No arguments

Return type

No return value

Example

To blank box b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boxs in model m:

```
Box.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be blanked in
flag	Flag	Flag set on the boxs that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boxs in model m flagged with f:

```
Box.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the box is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if box b is blanked:

```
if (b.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse box b:

```
b.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the box.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the box

Return type

No return value

Example

To clear flag f for box b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the box.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Box object

Example

To copy box b into box z:

```
var z = b.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a box.

Arguments

Name	Type	Description
Model	Model	Model that the box will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Box object (or null if not made)

Example

To start creating a box in model m:

```
var m = Box.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a box.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the box

Return type

No return value

Example

To detach comment c from the box b:

```
b.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit box b:

```
b.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for box. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for box b:

```
b.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first box in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first box in

Return type

Box object (or null if there are no boxs in the model).

Example

To get the first box in model m:

```
var b = Box.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free box label in the model. Also see Box.LastFreeLabel(), Box.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free box label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Box label.

Example

To get the first free box label in model m:

```
var label = Box.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the boxs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be flagged in
flag	Flag	Flag to set on the boxs

Return type

No return value

Example

To flag all of the boxs with flag f in model m:

```
Box.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the box is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the box

Return type

true if flagged, false if not.

Example

To check if box b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each box in the model.
Note that ForEach has been designed to make looping over boxs as fast as possible and so has some limitations. Firstly, a single temporary Box object is created and on each function call it is updated with the current box data. This means that you should not try to store the Box object for later use (e.g. in an array) as it is temporary. Secondly, you cannot create new boxs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all boxs are in
func	function	Function to call for each box
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the boxs in model m:

```
Box.ForEach(m, test);
function test(b)
{
// b is Box object
}
```

To call function test for all of the boxs in model m with optional object:

```
var data = { x:0, y:0 };
Box.ForEach(m, test, data);
function test(b, extra)
{
// b is Box object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Box objects for all of the boxs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boxs from

Return type

Array of Box objects

Example

To make an array of Box objects for all of the boxs in model m

```
var b = Box.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a box.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the box b:

```
var comm_array = b.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Box objects for all of the flagged boxs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boxs from
flag	Flag	Flag set on the boxs that you want to retrieve

Return type

Array of Box objects

Example

To make an array of Box objects for all of the boxs in model m flagged with f

```
var b = Box.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Box object for a box ID.

Arguments

Name	Type	Description
Model	Model	Model to find the box in
number	integer	number of the box you want the Box object for

Return type

Box object (or null if box does not exist).

Example

To get the Box object for box 100 in model m

```
var b = Box.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Box property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Box.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	box property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Box property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Box property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this box (*DEFINE_BOX). **Note that a carriage return is not added**. See also Box.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for box m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the box. **Note that a carriage return is not added**. See also Box.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for box b:

```
var cards = b.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last box in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last box in

Return type

Box object (or null if there are no boxs in the model).

Example

To get the last box in model m:

```
var b = Box.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free box label in the model. Also see Box.FirstFreeLabel(), Box.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free box label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Box label.

Example

To get the last free box label in model m:

```
var label = Box.LastFreeLabel(m);
```

Next()

Description

Returns the next box in the model.

Arguments

No arguments

Return type

Box object (or null if there are no more boxs in the model).

Example

To get the box in model m after box b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) box label in the model. Also see Box.FirstFreeLabel(), Box.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free box label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Box label.

Example

To get the next free box label in model m:

```
var label = Box.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a box.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boxs from that model can be picked. If the argument is a Flag then only boxs that are flagged with *limit* can be selected. If omitted, or null, any boxs from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Box object (or null if not picked)

Example

To pick a box from model m giving the prompt 'Pick box from screen':

```
var b = Box.Pick('Pick box from screen', m);
```

Previous()

Description

Returns the previous box in the model.

Arguments

No arguments

Return type

Box object (or null if there are no more boxs in the model).

Example

To get the box in model m before box b:

```
var b = b.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boxs in model m, from 1000000:

```
Box.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be renumbered in
flag	Flag	Flag set on the boxs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boxs in model m flagged with f, from 1000000:

```
Box.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select boxs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting boxs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boxs from that model can be selected. If the argument is a Flag then only boxs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any boxs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of boxs selected or null if menu cancelled

Example

To select boxs from model m, flagging those selected with flag f, giving the prompt 'Select boxs':

```
Box.Select(f, 'Select boxs', m);
```

To select boxs, flagging those selected with flag f but limiting selection to boxs flagged with flag l, giving the prompt 'Select boxs':

```
Box.Select(f, 'Select boxs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the box.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the box

Return type

No return value

Example

To set flag f for box b:

```
b.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the box. The box will be sketched until you either call Box.Unsketch(), Box.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the box is sketched. If omitted redraw is true. If you want to sketch several boxs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch box b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged boxs in the model. The boxs will be sketched until you either call Box.Unsketch(), Box.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be sketched in
flag	Flag	Flag set on the boxs that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the boxs are sketched. If omitted redraw is true. If you want to sketch flagged boxs several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all boxs flagged with flag in model m:

```
Box.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing boxs should be counted. If false or omitted referenced but undefined boxs will also be included in the total.

Return type

number of boxs

Example

To get the total number of boxs in model m:

```
var total = Box.Total(m);
```

Unblank()

Description

Unblanks the box

Arguments

No arguments

Return type

No return value

Example

To unblank box b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boxs in model m:

```
Box.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged boxs will be unblanked in
flag	Flag	Flag set on the boxs that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boxs in model m flagged with f:

```
Box.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all boxs will be unset in
flag	Flag	Flag to unset on the boxs

Return type

No return value

Example

To unset the flag f on all the boxs in model m:

```
Box.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the box.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the box is unsketched. If omitted redraw is true. If you want to unsketch several boxs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch box b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all boxs.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the boxs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boxs in model m:

```
Box.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unsketched in
flag	Flag	Flag set on the boxs that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the boxs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boxs flagged with flag in model m:

```
Box.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Box object.

Example

To check if Box property b.example is a parameter by using the Box.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for box. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for box b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this box.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for box b:

```
var xrefs = b.Xrefs();
```

---

toString()

Description

Creates a string containing the box data in keyword format. Note that this contains the keyword header and the keyword cards. See also Box.Keyword() and Box.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for box b in keyword format

```
var s = b.toString();
```

ConnectionProperties class

The ConnectionProperties class gives you access to *DEFINE_CONNECTION_PROPERTIES keyword in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AddMaterialDataLine()
- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetMaterialDataLine(row*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveMaterialDataLine(row*[integer]*)
- SetFlag(flag*[Flag]*)
- SetMaterialDataLine(row*[integer]*, mid*[integer]*, sigy (optional)*[real]*, etan (optional)*[real]*, dg_pr (optional)*[real]*, rank (optional)*[real]*, sn (optional)*[real]*, sb (optional)*[real]*, ss (optional)*[real]*, exsn (optional)*[real]*, exsb (optional)*[real]*, exss (optional)*[real]*, lcsn (optional)*[integer]*, lcsb (optional)*[integer]*, lcss (optional)*[integer]*, gfad (optional)*[real]*, sclmrr (optional)*[real]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ConnectionProperties properties

Name	Type	Description
add	integer	To _ADD case's parent definition
areaeq	integer	Area equation number.
con_id	integer	*DEFINE_CONNECTION_PROPERTIES id.
d_dg_pr	real	Default damage parameter.
d_dg_prf	integer	Default damage parameter(function if proprul == 2).
d_etan	real	Default tangent modulus.
d_etanf	integer	Default tangent modulus(function if proprul == 2).
d_exsb	real	Default bending stress exponent.
d_exsbf	integer	Default bending stress exponent(function if proprul == 2).
d_exsn	real	Default normal stress exponent.
d_exsnf	integer	Default normal stress exponent(function if proprul == 2).
d_exss	real	Default shear stress exponent.
d_exssf	integer	Default shear stress exponent(function if proprul == 2).
d_gfad	real	Default fading energy.
d_gfadf	integer	Default fading energy(function if proprul == 2).
d_lcsb	integer	Default LC of bending stress scale factor wrt strain rate.
d_lcsn	integer	Default LC of normal stress scale factor wrt strain rate.
d_lcss	integer	Default LC of shear stress scale factor wrt strain rate.
d_rank	real	Default rank value.
d_sb	real	Default bending strength.
d_sbf	integer	Default bending strength(function if proprul == 2).
d_sclmrr	real	Default scaling factor for torsional moment in failure function.
d_sigy	real	Default yield stress.
d_sigyf	integer	Default yield stress(function if proprul == 2).
d_sn	real	Default normal strength.
d_snf	integer	Default normal strength(function if proprul == 2).
d_ss	real	Default shear strength.
d_ssf	integer	Default shear strength(function if proprul == 2).
dg_typ	integer	Damage type.
exists	logical	true if *DEFINE_CONNECTION_PROPERTIES exists, false if referred to but not defined. (read only)
heading	string	The title of the *DEFINE_CONNECTION_PROPERTIES or the empty string if _TITLE is not set
include	integer	The Include file number that the *DEFINE_CONNECTION_PROPERTIES is in.
moarfl	integer	Modelled area flag.
model	integer	The Model number that the *DEFINE_CONNECTION_PROPERTIES is in.
proprul	integer	Property rule number.

Detailed Description

The ConnectionProperties class allows you to create, modify, edit and manipulate *DEFINE_CONNECTION_PROPERTIES. See the documentation below for more details.

Constructor

new ConnectionProperties(Model*[Model]*, con_id*[integer]*, heading (optional)*[string]*)

Description

Create a new *DEFINE_CONNECTION_PROPERTIES object.

Arguments

Name	Type	Description
Model	Model	Model that *DEFINE_CONNECTION_PROPERTIES will be created in
con_id	integer	*DEFINE_CONNECTION_PROPERTIES id.
heading (optional)	string	Title for the *DEFINE_CONNECTION_PROPERTIES

Return type

ConnectionProperties object

Example

To create a new *DEFINE_CONNECTION_PROPERTIES in model m with label 100:

```
var c = new ConnectionProperties(m, 100);
```

Details of functions

AddMaterialDataLine()

Description

Allows user to add material data line in *DEFINE_CONNECTION_PROPERTIES.

Arguments

No arguments

Return type

No return value

Example

To Add Material data line in *DEFINE_CONNECTION_PROPERTIES c:

```
c.AddMaterialDataLine();
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the *DEFINE_CONNECTION_PROPERTIES

Return type

No return value

Example

To associate comment c to the *DEFINE_CONNECTION_PROPERTIES c:

```
c.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse *DEFINE_CONNECTION_PROPERTIES c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the *DEFINE_CONNECTION_PROPERTIES

Return type

No return value

Example

To clear flag f for *DEFINE_CONNECTION_PROPERTIES c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ConnectionProperties object

Example

To copy *DEFINE_CONNECTION_PROPERTIES c into *DEFINE_CONNECTION_PROPERTIES z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
Model	Model	Model that the *DEFINE_CONNECTION_PROPERTIES will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ConnectionProperties object (or null if not made)

Example

To start creating a *DEFINE_CONNECTION_PROPERTIES in model m:

```
var c = ConnectionProperties.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the *DEFINE_CONNECTION_PROPERTIES

Return type

No return value

Example

To detach comment c from the *DEFINE_CONNECTION_PROPERTIES c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit *DEFINE_CONNECTION_PROPERTIES c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for *DEFINE_CONNECTION_PROPERTIES. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for *DEFINE_CONNECTION_PROPERTIES c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first *DEFINE_CONNECTION_PROPERTIES in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first *DEFINE_CONNECTION_PROPERTIES in

Return type

ConnectionProperties object (or null if there are no *DEFINE_CONNECTION_PROPERTIESs in the model).

Example

To get the first *DEFINE_CONNECTION_PROPERTIES in model m:

```
var c = ConnectionProperties.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free *DEFINE_CONNECTION_PROPERTIES label in the model. Also see ConnectionProperties.LastFreeLabel(), ConnectionProperties.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free *DEFINE_CONNECTION_PROPERTIES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

ConnectionProperties label.

Example

To get the first free *DEFINE_CONNECTION_PROPERTIES label in model m:

```
var label = ConnectionProperties.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the *DEFINE_CONNECTION_PROPERTIESs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONNECTION_PROPERTIESs will be flagged in
flag	Flag	Flag to set on the *DEFINE_CONNECTION_PROPERTIESs

Return type

No return value

Example

To flag all of the *DEFINE_CONNECTION_PROPERTIESs with flag f in model m:

```
ConnectionProperties.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the *DEFINE_CONNECTION_PROPERTIES is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the *DEFINE_CONNECTION_PROPERTIES

Return type

true if flagged, false if not.

Example

To check if *DEFINE_CONNECTION_PROPERTIES c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each *DEFINE_CONNECTION_PROPERTIES in the model.
Note that ForEach has been designed to make looping over *DEFINE_CONNECTION_PROPERTIESs as fast as possible and so has some limitations.
Firstly, a single temporary ConnectionProperties object is created and on each function call it is updated with the current *DEFINE_CONNECTION_PROPERTIES data. This means that you should not try to store the ConnectionProperties object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new *DEFINE_CONNECTION_PROPERTIESs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONNECTION_PROPERTIESs are in
func	function	Function to call for each *DEFINE_CONNECTION_PROPERTIES
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the *DEFINE_CONNECTION_PROPERTIESs in model m:

```
ConnectionProperties.ForEach(m, test);
function test(c)
{
// c is ConnectionProperties object
}
```

To call function test for all of the *DEFINE_CONNECTION_PROPERTIESs in model m with optional object:

```
var data = { x:0, y:0 };
ConnectionProperties.ForEach(m, test, data);
function test(c, extra)
{
// c is ConnectionProperties object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of ConnectionProperties objects for all of the *DEFINE_CONNECTION_PROPERTIESs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *DEFINE_CONNECTION_PROPERTIESs from

Return type

Array of ConnectionProperties objects

Example

To make an array of ConnectionProperties objects for all of the *DEFINE_CONNECTION_PROPERTIESs in model m

```
var c = ConnectionProperties.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a *DEFINE_CONNECTION_PROPERTIES.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the *DEFINE_CONNECTION_PROPERTIES c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of ConnectionProperties objects for all of the flagged *DEFINE_CONNECTION_PROPERTIESs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *DEFINE_CONNECTION_PROPERTIESs from
flag	Flag	Flag set on the *DEFINE_CONNECTION_PROPERTIESs that you want to retrieve

Return type

Array of ConnectionProperties objects

Example

To make an array of ConnectionProperties objects for all of the *DEFINE_CONNECTION_PROPERTIESs in model m flagged with f

```
var c = ConnectionProperties.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ConnectionProperties object for a *DEFINE_CONNECTION_PROPERTIES ID.

Arguments

Name	Type	Description
Model	Model	Model to find the *DEFINE_CONNECTION_PROPERTIES in
number	integer	number of the *DEFINE_CONNECTION_PROPERTIES you want the ConnectionProperties object for

Return type

ConnectionProperties object (or null if *DEFINE_CONNECTION_PROPERTIES does not exist).

Example

To get the ConnectionProperties object for *DEFINE_CONNECTION_PROPERTIES 100 in model m

```
var c = ConnectionProperties.GetFromID(m, 100);
```

GetMaterialDataLine(row*[integer]*)

Description

Returns the material data at given row in *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
row	integer	Material data row number, eg. for first material data, row = 0

Return type

Array of numbers containing the material id, sigy, e_tan etc. .

Example

To get material data at first row, row = 0 in *DEFINE_CONNECTION_PROPERTIES c:

```
c.GetMaterialData(0);
```

GetParameter(prop*[string]*)

Description

Checks if a ConnectionProperties property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ConnectionProperties.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	*DEFINE_CONNECTION_PROPERTIES property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ConnectionProperties property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ConnectionProperties property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this *DEFINE_CONNECTION_PROPERTIES **Note that a carriage return is not added**. See also ConnectionProperties.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for ConnectionProperties c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the *DEFINE_CONNECTION_PROPERTIES. **Note that a carriage return is not added**. See also ConnectionProperties.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for connection_properties c:

```
var cards = c.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last *DEFINE_CONNECTION_PROPERTIES in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last *DEFINE_CONNECTION_PROPERTIES in

Return type

ConnectionProperties object (or null if there are no *DEFINE_CONNECTION_PROPERTIESs in the model).

Example

To get the last *DEFINE_CONNECTION_PROPERTIES in model m:

```
var c = ConnectionProperties.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free *DEFINE_CONNECTION_PROPERTIES label in the model. Also see ConnectionProperties.FirstFreeLabel(), ConnectionProperties.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free *DEFINE_CONNECTION_PROPERTIES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

ConnectionProperties label.

Example

To get the last free *DEFINE_CONNECTION_PROPERTIES label in model m:

```
var label = ConnectionProperties.LastFreeLabel(m);
```

Next()

Description

Returns the next *DEFINE_CONNECTION_PROPERTIES in the model.

Arguments

No arguments

Return type

ConnectionProperties object (or null if there are no more *DEFINE_CONNECTION_PROPERTIESs in the model).

Example

To get the *DEFINE_CONNECTION_PROPERTIES in model m after *DEFINE_CONNECTION_PROPERTIES c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) *DEFINE_CONNECTION_PROPERTIES label in the model. Also see ConnectionProperties.FirstFreeLabel(), ConnectionProperties.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free *DEFINE_CONNECTION_PROPERTIES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

ConnectionProperties label.

Example

To get the next free *DEFINE_CONNECTION_PROPERTIES label in model m:

```
var label = ConnectionProperties.NextFreeLabel(m);
```

Previous()

Description

Returns the previous *DEFINE_CONNECTION_PROPERTIES in the model.

Arguments

No arguments

Return type

ConnectionProperties object (or null if there are no more *DEFINE_CONNECTION_PROPERTIESs in the model).

Example

To get the *DEFINE_CONNECTION_PROPERTIES in model m before *DEFINE_CONNECTION_PROPERTIES c:

```
var c = c.Previous();
```

RemoveMaterialDataLine(row*[integer]*)

Description

Allows user to remove material data line in *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
row	integer	Material data row number, eg. for first material data, row = 0

Return type

No return value

Example

To remove first material data line in *DEFINE_CONNECTION_PROPERTIES c:

```
c.RemoveMaterialDataLine(0);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the *DEFINE_CONNECTION_PROPERTIESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONNECTION_PROPERTIESs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *DEFINE_CONNECTION_PROPERTIESs in model m, from 1000000:

```
ConnectionProperties.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged *DEFINE_CONNECTION_PROPERTIESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *DEFINE_CONNECTION_PROPERTIESs will be renumbered in
flag	Flag	Flag set on the *DEFINE_CONNECTION_PROPERTIESs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *DEFINE_CONNECTION_PROPERTIESs in model m flagged with f, from 1000000:

```
ConnectionProperties.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select *DEFINE_CONNECTION_PROPERTIESs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting *DEFINE_CONNECTION_PROPERTIESs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only *DEFINE_CONNECTION_PROPERTIESs from that model can be selected. If the argument is a Flag then only *DEFINE_CONNECTION_PROPERTIESs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any *DEFINE_CONNECTION_PROPERTIESs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of *DEFINE_CONNECTION_PROPERTIESs selected or null if menu cancelled

Example

To select *DEFINE_CONNECTION_PROPERTIESs from model m, flagging those selected with flag f, giving the prompt 'Select *DEFINE_CONNECTION_PROPERTIESs':

```
ConnectionProperties.Select(f, 'Select *DEFINE_CONNECTION_PROPERTIESs', m);
```

To select *DEFINE_CONNECTION_PROPERTIESs, flagging those selected with flag f but limiting selection to *DEFINE_CONNECTION_PROPERTIESs flagged with flag l, giving the prompt 'Select *DEFINE_CONNECTION_PROPERTIESs':

```
ConnectionProperties.Select(f, 'Select *DEFINE_CONNECTION_PROPERTIESs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the *DEFINE_CONNECTION_PROPERTIES

Return type

No return value

Example

To set flag f for *DEFINE_CONNECTION_PROPERTIES c:

```
c.SetFlag(f);
```

SetMaterialDataLine(row*[integer]*, mid*[integer]*, sigy (optional)*[real]*, etan (optional)*[real]*, dg_pr (optional)*[real]*, rank (optional)*[real]*, sn (optional)*[real]*, sb (optional)*[real]*, ss (optional)*[real]*, exsn (optional)*[real]*, exsb (optional)*[real]*, exss (optional)*[real]*, lcsn (optional)*[integer]*, lcsb (optional)*[integer]*, lcss (optional)*[integer]*, gfad (optional)*[real]*, sclmrr (optional)*[real]*)

Description

Allows user to set fields for material data line at given row in *DEFINE_CONNECTION_PROPERTIES.

Arguments

Name	Type	Description
row	integer	Material data row number, eg. for first material data, row = 0
mid	integer	Material ID
sigy (optional)	real	Default yield stress
etan (optional)	real	Default tangent modulus
dg_pr (optional)	real	Default damage parameter
rank (optional)	real	Default rank value
sn (optional)	real	Default normal strength
sb (optional)	real	Default bending strength
ss (optional)	real	Default shear strength
exsn (optional)	real	Default normal stress exponent
exsb (optional)	real	Default bending stress exponent
exss (optional)	real	Default shear stress exponent
lcsn (optional)	integer	Default LC of normal stress scale factor wrt strain rate
lcsb (optional)	integer	Default LC of bending stress scale factor wrt strain rate
lcss (optional)	integer	Default LC of shear stress scale factor wrt strain rate
gfad (optional)	real	Default fading energy
sclmrr (optional)	real	Default scaling factor for torsional moment in failure function

Return type

No return value

Example

To set material data at first row (row = 0) to mat 111 in *DEFINE_CONNECTION_PROPERTIES c:

```
c.SetMaterialData(0,111);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of *DEFINE_CONNECTION_PROPERTIESs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing *DEFINE_CONNECTION_PROPERTIESs should be counted. If false or omitted referenced but undefined *DEFINE_CONNECTION_PROPERTIESs will also be included in the total.

Return type

number of *DEFINE_CONNECTION_PROPERTIESs

Example

To get the total number of *DEFINE_CONNECTION_PROPERTIESs in model m:

```
var total = ConnectionProperties.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the *DEFINE_CONNECTION_PROPERTIESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all *DEFINE_CONNECTION_PROPERTIESs will be unset in
flag	Flag	Flag to unset on the *DEFINE_CONNECTION_PROPERTIESs

Return type

No return value

Example

To unset the flag f on all the *DEFINE_CONNECTION_PROPERTIESs in model m:

```
ConnectionProperties.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ConnectionProperties object.

Example

To check if ConnectionProperties property c.example is a parameter by using the ConnectionProperties.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for *DEFINE_CONNECTION_PROPERTIES. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for *DEFINE_CONNECTION_PROPERTIES c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this *DEFINE_CONNECTION_PROPERTIES.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for *DEFINE_CONNECTION_PROPERTIES c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the connection_properties data in keyword format. Note that this contains the keyword header and the keyword cards. See also ConnectionProperties.Keyword() and ConnectionProperties.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for connection_properties c in keyword format

```
var str = c.toString();
```

ConstructionStages class

The ConstructionStages class gives you access to *DEFINE_CONSTRUCTION_STAGES keyword in PRIMER.
More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ConstructionStages properties

Name	Type	Description
ate	real	Analysis time at end of stage.
atr	real	Analysis time duration of stage.
ats	real	Analysis time at start of stage.

exists	logical	true if *DEFINE_CONSTRUCTION_STAGES exists, false if referred to but not defined. (read only)
heading	string	The title of the *DEFINE_CONSTRUCTION_STAGES or the empty string if _TITLE is not set
include	integer	The Include file number that the *DEFINE_CONSTRUCTION_STAGES is in.
istage	integer	ConstructionStages number. The label is an alternative name for this.
ivel0	integer	Flag to set velocities to zero at start of stage.
label	integer	ConstructionStages number. The istage is an alternative name for this.
model	integer	The Model number that the *DEFINE_CONSTRUCTION_STAGES is in.
rte	real	Real time at end of stage.
rts	real	Real time at start of stage.

Detailed Description

The ConstructionStages class allows you to create, modify, edit and manipulate *DEFINE_CONSTRUCTION_STAGES. See the documentation below for more details.

Constructor

new ConstructionStages(Model*[Model]*, Stage ID*[integer]*, heading (optional)*[string]*)

Description

Create a new ConstructionStages object.

Arguments

Name	Type	Description
Model	Model	Model that *DEFINE_CONSTRUCTION_STAGES will be created in
Stage ID	integer	ConstructionStages id.
heading (optional)	string	Title for the *DEFINE_CONSTRUCTION_STAGES

Return type

ConstructionStages object

Example

To create a new *DEFINE_CONSTRUCTION_STAGES in model m with label 100:

```
var c = new ConstructionStages(m, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be attached to the *DEFINE_CONSTRUCTION_STAGES

Return type

No return value

Example

To associate comment c to the *DEFINE_CONSTRUCTION_STAGES c:

```
c.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse *DEFINE_CONSTRUCTION_STAGES c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the *DEFINE_CONSTRUCTION_STAGES

Return type

No return value

Example

To clear flag f for *DEFINE_CONSTRUCTION_STAGES c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ConstructionStages object

Example

To copy *DEFINE_CONSTRUCTION_STAGES c into *DEFINE_CONSTRUCTION_STAGES z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description
Model	Model	Model that the *DEFINE_CONSTRUCTION_STAGES will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ConstructionStages object (or null if not made)

Example

To start creating a *DEFINE_CONSTRUCTION_STAGES in model m:

```
var c = ConstructionStages.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the *DEFINE_CONSTRUCTION_STAGES

Return type

No return value

Example

To detach comment c from the *DEFINE_CONSTRUCTION_STAGES c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit *DEFINE_CONSTRUCTION_STAGES c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for *DEFINE_CONSTRUCTION_STAGES. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for *DEFINE_CONSTRUCTION_STAGES c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first *DEFINE_CONSTRUCTION_STAGES in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first *DEFINE_CONSTRUCTION_STAGES in

Return type

ConstructionStages object (or null if there are no *DEFINE_CONSTRUCTION_STAGESs in the model).

Example

To get the first *DEFINE_CONSTRUCTION_STAGES in model m:

```
var c = ConstructionStages.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free *DEFINE_CONSTRUCTION_STAGES label in the model. Also see ConstructionStages.LastFreeLabel(), ConstructionStages.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free *DEFINE_CONSTRUCTION_STAGES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

ConstructionStages label.

Example

To get the first free *DEFINE_CONSTRUCTION_STAGES label in model m:

```
var label = ConstructionStages.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the *DEFINE_CONSTRUCTION_STAGESs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONSTRUCTION_STAGESs will be flagged in
flag	Flag	Flag to set on the *DEFINE_CONSTRUCTION_STAGESs

Return type

No return value

Example

To flag all of the *DEFINE_CONSTRUCTION_STAGESs with flag f in model m:

```
ConstructionStages.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the *DEFINE_CONSTRUCTION_STAGES is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the *DEFINE_CONSTRUCTION_STAGES

Return type

true if flagged, false if not.

Example

To check if *DEFINE_CONSTRUCTION_STAGES c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each *DEFINE_CONSTRUCTION_STAGES in the model.
Note that ForEach has been designed to make looping over *DEFINE_CONSTRUCTION_STAGESs as fast as possible and so has some limitations.
Firstly, a single temporary ConstructionStages object is created and on each function call it is updated with the current *DEFINE_CONSTRUCTION_STAGES data. This means that you should not try to store the ConstructionStages object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new *DEFINE_CONSTRUCTION_STAGESs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONSTRUCTION_STAGESs are in
func	function	Function to call for each *DEFINE_CONSTRUCTION_STAGES
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the *DEFINE_CONSTRUCTION_STAGESs in model m:

```
ConstructionStages.ForEach(m, test);
function test(c)
{
// c is ConstructionStages object
}
```

To call function test for all of the *DEFINE_CONSTRUCTION_STAGESs in model m with optional object:

```
var data = { x:0, y:0 };
ConstructionStages.ForEach(m, test, data);
function test(c, extra)
{
// c is ConstructionStages object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of ConstructionStages objects for all of the *DEFINE_CONSTRUCTION_STAGESs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *DEFINE_CONSTRUCTION_STAGESs from

Return type

Array of ConstructionStages objects

Example

To make an array of ConstructionStages objects for all of the *DEFINE_CONSTRUCTION_STAGESs in model m

```
var c = ConstructionStages.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a *DEFINE_CONSTRUCTION_STAGES.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the *DEFINE_CONSTRUCTION_STAGES c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of ConstructionStages objects for all of the flagged *DEFINE_CONSTRUCTION_STAGESs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *DEFINE_CONSTRUCTION_STAGESs from
flag	Flag	Flag set on the *DEFINE_CONSTRUCTION_STAGESs that you want to retrieve

Return type

Array of ConstructionStages objects

Example

To make an array of ConstructionStages objects for all of the *DEFINE_CONSTRUCTION_STAGESs in model m flagged with f

```
var c = ConstructionStages.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ConstructionStages object for a *DEFINE_CONSTRUCTION_STAGES ID.

Arguments

Name	Type	Description
Model	Model	Model to find the *DEFINE_CONSTRUCTION_STAGES in
number	integer	number of the *DEFINE_CONSTRUCTION_STAGES you want the ConstructionStages object for

Return type

ConstructionStages object (or null if *DEFINE_CONSTRUCTION_STAGES does not exist).

Example

To get the ConstructionStages object for *DEFINE_CONSTRUCTION_STAGES 100 in model m

```
var c = ConstructionStages.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a ConstructionStages property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ConstructionStages.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	*DEFINE_CONSTRUCTION_STAGES property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ConstructionStages property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ConstructionStages property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this *DEFINE_CONSTRUCTION_STAGES. **Note that a carriage return is not added**. See also ConstructionStages.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for ConstructionStages c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the *DEFINE_CONSTRUCTION_STAGES. **Note that a carriage return is not added**. See also ConstructionStages.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for construction_stages c:

```
var cards = c.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last *DEFINE_CONSTRUCTION_STAGES in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last *DEFINE_CONSTRUCTION_STAGES in

Return type

ConstructionStages object (or null if there are no *DEFINE_CONSTRUCTION_STAGESs in the model).

Example

To get the last *DEFINE_CONSTRUCTION_STAGES in model m:

```
var c = ConstructionStages.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free *DEFINE_CONSTRUCTION_STAGES label in the model. Also see ConstructionStages.FirstFreeLabel(), ConstructionStages.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free *DEFINE_CONSTRUCTION_STAGES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

ConstructionStages label.

Example

To get the last free *DEFINE_CONSTRUCTION_STAGES label in model m:

```
var label = ConstructionStages.LastFreeLabel(m);
```

Next()

Description

Returns the next *DEFINE_CONSTRUCTION_STAGES in the model.

Arguments

No arguments

Return type

ConstructionStages object (or null if there are no more *DEFINE_CONSTRUCTION_STAGESs in the model).

Example

To get the *DEFINE_CONSTRUCTION_STAGES in model m after *DEFINE_CONSTRUCTION_STAGES c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) *DEFINE_CONSTRUCTION_STAGES label in the model. Also see ConstructionStages.FirstFreeLabel(), ConstructionStages.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free *DEFINE_CONSTRUCTION_STAGES label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

ConstructionStages label.

Example

To get the next free *DEFINE_CONSTRUCTION_STAGES label in model m:

```
var label = ConstructionStages.NextFreeLabel(m);
```

Previous()

Description

Returns the previous *DEFINE_CONSTRUCTION_STAGES in the model.

Arguments

No arguments

Return type

ConstructionStages object (or null if there are no more *DEFINE_CONSTRUCTION_STAGESs in the model).

Example

To get the *DEFINE_CONSTRUCTION_STAGES in model m before *DEFINE_CONSTRUCTION_STAGES c:

```
var c = c.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the *DEFINE_CONSTRUCTION_STAGESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all *DEFINE_CONSTRUCTION_STAGESs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *DEFINE_CONSTRUCTION_STAGESs in model m, from 1000000:

```
ConstructionStages.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged *DEFINE_CONSTRUCTION_STAGESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *DEFINE_CONSTRUCTION_STAGESs will be renumbered in
flag	Flag	Flag set on the *DEFINE_CONSTRUCTION_STAGESs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *DEFINE_CONSTRUCTION_STAGESs in model m flagged with f, from 1000000:

```
ConstructionStages.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select *DEFINE_CONSTRUCTION_STAGESs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting *DEFINE_CONSTRUCTION_STAGESs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only *DEFINE_CONSTRUCTION_STAGESs from that model can be selected. If the argument is a Flag then only *DEFINE_CONSTRUCTION_STAGESs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any *DEFINE_CONSTRUCTION_STAGESs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of *DEFINE_CONSTRUCTION_STAGESs selected or null if menu cancelled

Example

To select *DEFINE_CONSTRUCTION_STAGESs from model m, flagging those selected with flag f, giving the prompt 'Select *DEFINE_CONSTRUCTION_STAGESs':

```
ConstructionStages.Select(f, 'Select *DEFINE_CONSTRUCTION_STAGESs', m);
```

To select *DEFINE_CONSTRUCTION_STAGESs, flagging those selected with flag f but limiting selection to *DEFINE_CONSTRUCTION_STAGESs flagged with flag l, giving the prompt 'Select *DEFINE_CONSTRUCTION_STAGESs':

```
ConstructionStages.Select(f, 'Select *DEFINE_CONSTRUCTION_STAGESs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the *DEFINE_CONSTRUCTION_STAGES.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the *DEFINE_CONSTRUCTION_STAGES

Return type

No return value

Example

To set flag f for *DEFINE_CONSTRUCTION_STAGES c:

```
c.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of *DEFINE_CONSTRUCTION_STAGESs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing *DEFINE_CONSTRUCTION_STAGESs should be counted. If false or omitted referenced but undefined *DEFINE_CONSTRUCTION_STAGESs will also be included in the total.

Return type

number of *DEFINE_CONSTRUCTION_STAGESs

Example

To get the total number of *DEFINE_CONSTRUCTION_STAGESs in model m:

```
var total = ConstructionStages.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the *DEFINE_CONSTRUCTION_STAGESs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all *DEFINE_CONSTRUCTION_STAGESs will be unset in
flag	Flag	Flag to unset on the *DEFINE_CONSTRUCTION_STAGESs

Return type

No return value

Example

To unset the flag f on all the *DEFINE_CONSTRUCTION_STAGESs in model m:

```
ConstructionStages.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ConstructionStages object.

Example

To check if ConstructionStages property c.example is a parameter by using the ConstructionStages.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for *DEFINE_CONSTRUCTION_STAGES. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for *DEFINE_CONSTRUCTION_STAGES c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this *DEFINE_CONSTRUCTION_STAGES.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for *DEFINE_CONSTRUCTION_STAGES c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the construction stages data in keyword format. Note that this contains the keyword header and the keyword cards. See also ConstructionStages.Keyword() and ConstructionStages.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for construction stages c in keyword format

```
var str = c.toString();
```

CoordinateSystem (Csys) class

The CoordinateSystem class gives you access to define coordinate cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

CoordinateSystem constants

Name	Description
CoordinateSystem.NODES	Csys is *DEFINE_COORDINATE_NODES.
CoordinateSystem.SYSTEM	Csys is *DEFINE_COORDINATE_SYSTEM.
CoordinateSystem.VECTOR	Csys is *DEFINE_COORDINATE_VECTOR.

CoordinateSystem properties

Name	Type	Description
cid	integer	CoordinateSystem number. Also see the label number.
cidl	integer	Optional local coordinate system to define the points in
dir	int	Axis defined by N1N2
exists	logical	true if csys exists, false if referred to but not defined. (read only)
flag	logical	Flag for updating local system each timestep
heading	string	CoordinateSystem heading
include	integer	The Include file number that the csys is in.
label	integer	CoordinateSystem number. Also see the cid property which is an alternative name for this.
lx	real	X-coordinate of point on local X-axis
ly	real	Y-coordinate of point on local X-axis
lz	real	Z-coordinate of point on local X-axis
model	integer	The Model number that the coordinate system is in.
n1	int	Node located at local origin
n2	int	Node located along local (dir) axis
n3	int	Node located in local plane determined by (dir)
nid	integer	Optional node id for rotation
option	constant	CoordinateSystem type (Can be CoordinateSystem.NODES, CoordinateSystem.SYSTEM or CoordinateSystem.VECTOR).
ox	real	X-coordinate of origin
oy	real	Y-coordinate of origin
oz	real	Z-coordinate of origin
px	real	X-coordinate of point in local X-Y plane
py	real	Y-coordinate of point in local X-Y plane
pz	real	Z-coordinate of point in local X-Y plane
vx	real	X-coordinate of local X-Y vector
vy	real	Y-coordinate of local X-Y vector
vz	real	Z-coordinate of local X-Z vector
xx	real	X-coordinate on local X-axis
xy	real	Y-coordinate on local X-axis
xz	real	Z-coordinate on local X-axis

Detailed Description

The CoordinateSystem class allows you to create, modify, edit and manipulate csys cards. See the documentation below for more details.

For convenience "Csys" can also be used as the class name instead of "CoordinateSystem".

Constructor

new CoordinateSystem(Model*[Model]*, option*[constant]*, cid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, flag*[boolean]*, dir*[integer]*, heading (optional)*[string]*)

Description

Create a new CoordinateSystem object for *DEFINE_COORDINATE_NODES.

Arguments

Name	Type	Description
Model	Model	Model that csys will be created in
option	constant	Must be CoordinateSystem.NODES
cid	integer	CoordinateSystem number
n1	integer	Node located at origin
n2	integer	Node located along (DIR) axis
n3	integer	Node located in plane defined by (DIR)
flag	boolean	Flag for local system update each time step
dir	integer	Axis defined by N1N2
heading (optional)	string	Title for the csys

Return type

CoordinateSystem object

Example

To create a new Csys of type Nodes in model m with label 200 and title "Test csys 1" defined by nodes 1, 2, 3 with where N1N2 defines local Y-axis; local system update flag is off

```
var c = new CoordinateSystem(m, CoordinateSystem.NODES, 200, 1, 2, 3, 0, 2,
"Test csys");
```

new CoordinateSystem(Model*[Model]*, option*[constant]*, cid*[integer]*, ox*[real]*, oy*[real]*, oz*[real]*, lx*[real]*, ly*[real]*, lz*[real]*, px*[real]*, py*[real]*, pz*[real]*, heading (optional)*[string]*)

Description

Create a new CoordinateSystem object for *DEFINE_COORDINATE_SYSTEM.

Arguments

Name	Type	Description
Model	Model	Model that csys will be created in
option	constant	Must be CoordinateSystem.SYSTEM

cid	integer	CoordinateSystem number
ox	real	X-coordinate of origin
oy	real	Y-coordinate of origin
oz	real	Z-coordinate of origin
lx	real	X-coordinate of point on local X-axis
ly	real	Y-coordinate of point on local X-axis
lz	real	Z-coordinate of point on local X-axis
px	real	X-coordinate of point in local X-Y plane
py	real	Y-coordinate of point in local X-Y plane
pz	real	Z-coordinate of point in local X-Y plane
heading (optional)	string	Title for the csys

Return type

No return value

Example

To create a new Csys of type Points in model m with label 300 and title "Test csys 2" with origin at (10, 10, 0), point on local X-axis at (20, 20, 0) and point on X-y at (10, 20, 0)

```
var c = new CoordinateSystem(m, CoordinateSystem.SYSTEM, 300, 10, 10, 0, 20, 20, 0, 10, 20, 0, "Test csys");
```

new CoordinateSystem(Model*[Model]*, option*[constant]*, cid*[integer]*, xx*[real]*, xy*[real]*, xz*[real]*, vx*[real]*, vy*[real]*, vz*[real]*, nid*[integer]*)

Description

Create a new CoordinateSystem object for *DEFINE_COORDINATE_VECTOR.

Arguments

Name	Type	Description
Model	Model	Model that csys will be created in
option	constant	Must be CoordinateSystem.VECTOR
cid	integer	CoordinateSystem number
xx	real	X-coordinate on local X-axis
xy	real	Y-coordinate on local X-axis
xz	real	Z-coordinate on local X-axis
vx	real	X-coordinate of local X-Y vector
vy	real	Y-coordinate of local X-Y vector
vz	real	Z-coordinate of local X-Z vector
nid	integer	Optional node id for rotation

Return type

No return value

Example

To create a new Csys of type Vectors in model m with label 400 with point on local X-axis at (50, 50, 0) and local XY being (-10, -20, 0) that can rotate with node 10003

```
var c = new CoordinateSystem(m, CoordinateSystem.VECTOR, 400, 50, 50, 0, -10,
-10, 10003);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a coordinate system.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the coordinate system

Return type

No return value

Example

To associate comment c to the coordinate system c:

```
c.AssociateComment(c);
```

Blank()

Description

Blanks the coordinate system

Arguments

No arguments

Return type

No return value

Example

To blank coordinate system c:

```
c.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the coordinate systems in model m:

```
CoordinateSystem.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged coordinate systems will be blanked in
flag	Flag	Flag set on the coordinate systems that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the coordinate systems in model m flagged with f:

```
CoordinateSystem.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the coordinate system is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if coordinate system c is blanked:

```
if (c.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse coordinate system c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the coordinate system.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the coordinate system

Return type

No return value

Example

To clear flag f for coordinate system c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the coordinate system.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

CoordinateSystem object

Example

To copy coordinate system c into coordinate system z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a csys.

Arguments

Name	Type	Description
Model	Model	Model that the csys will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

CoordinateSystem object (or null if not made)

Example

To start creating a csys in model m:

```
var m = CoordinateSystem.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a coordinate system.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the coordinate system

Return type

No return value

Example

To detach comment c from the coordinate system c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit coordinate system c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for coordinate system. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for coordinate system c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first coordinate system in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first coordinate system in

Return type

CoordinateSystem object (or null if there are no coordinate systems in the model).

Example

To get the first coordinate system in model m:

```
var c = CoordinateSystem.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free coordinate system label in the model. Also see CoordinateSystem.LastFreeLabel(), CoordinateSystem.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free coordinate system label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

CoordinateSystem label.

Example

To get the first free coordinate system label in model m:

```
var label = CoordinateSystem.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the coordinate systems in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems will be flagged in
flag	Flag	Flag to set on the coordinate systems

Return type

No return value

Example

To flag all of the coordinate systems with flag f in model m:

```
CoordinateSystem.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the coordinate system is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the coordinate system

Return type

true if flagged, false if not.

Example

To check if coordinate system c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each coordinate system in the model.
Note that ForEach has been designed to make looping over coordinate systems as fast as possible and so has some limitations.
Firstly, a single temporary CoordinateSystem object is created and on each function call it is updated with the current coordinate system data. This means that you should not try to store the CoordinateSystem object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new coordinate systems inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems are in
func	function	Function to call for each coordinate system
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the coordinate systems in model m:

```
CoordinateSystem.ForEach(m, test);
function test(c)
{
// c is CoordinateSystem object
}
```

To call function test for all of the coordinate systems in model m with optional object:

```
var data = { x:0, y:0 };
CoordinateSystem.ForEach(m, test, data);
function test(c, extra)
{
// c is CoordinateSystem object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of CoordinateSystem objects for all of the coordinate systems in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get coordinate systems from

Return type

Array of CoordinateSystem objects

Example

To make an array of CoordinateSystem objects for all of the coordinate systems in model m

```
var c = CoordinateSystem.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a coordinate system.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the coordinate system c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of CoordinateSystem objects for all of the flagged coordinate systems in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get coordinate systems from
flag	Flag	Flag set on the coordinate systems that you want to retrieve

Return type

Array of CoordinateSystem objects

Example

To make an array of CoordinateSystem objects for all of the coordinate systems in model m flagged with f

```
var c = CoordinateSystem.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the CoordinateSystem object for a coordinate system ID.

Arguments

Name	Type	Description
Model	Model	Model to find the coordinate system in
number	integer	number of the coordinate system you want the CoordinateSystem object for

Return type

CoordinateSystem object (or null if coordinate system does not exist).

Example

To get the CoordinateSystem object for coordinate system 100 in model m

```
var c = CoordinateSystem.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a CoordinateSystem property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the CoordinateSystem.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	coordinate system property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if CoordinateSystem property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if CoordinateSystem property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this csys (*DEFINE_COORDINATE). **Note that a carriage return is not added**. See also CoordinateSystem.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for csys m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the csys. **Note that a carriage return is not added**. See also CoordinateSystem.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for csys c:

```
var cards = v.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last coordinate system in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last coordinate system in

Return type

CoordinateSystem object (or null if there are no coordinate systems in the model).

Example

To get the last coordinate system in model m:

```
var c = CoordinateSystem.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free coordinate system label in the model. Also see CoordinateSystem.FirstFreeLabel(), CoordinateSystem.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free coordinate system label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

CoordinateSystem label.

Example

To get the last free coordinate system label in model m:

```
var label = CoordinateSystem.LastFreeLabel(m);
```

Next()

Description

Returns the next coordinate system in the model.

Arguments

No arguments

Return type

CoordinateSystem object (or null if there are no more coordinate systems in the model).

Example

To get the coordinate system in model m after coordinate system c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) coordinate system label in the model. Also see CoordinateSystem.FirstFreeLabel(), CoordinateSystem.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free coordinate system label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

CoordinateSystem label.

Example

To get the next free coordinate system label in model m:

```
var label = CoordinateSystem.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a coordinate system.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only coordinate systems from that model can be picked. If the argument is a Flag then only coordinate systems that are flagged with *limit* can be selected. If omitted, or null, any coordinate systems from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

CoordinateSystem object (or null if not picked)

Example

To pick a coordinate system from model m giving the prompt 'Pick coordinate system from screen':

```
var c = CoordinateSystem.Pick('Pick coordinate system from screen', m);
```

Previous()

Description

Returns the previous coordinate system in the model.

Arguments

No arguments

Return type

CoordinateSystem object (or null if there are no more coordinate systems in the model).

Example

To get the coordinate system in model m before coordinate system c:

```
var c = c.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the coordinate systems in model m, from 1000000:

```
CoordinateSystem.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged coordinate systems will be renumbered in
flag	Flag	Flag set on the coordinate systems that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the coordinate systems in model m flagged with f, from 1000000:

```
CoordinateSystem.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select coordinate systems using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting coordinate systems
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only coordinate systems from that model can be selected. If the argument is a Flag then only coordinate systems that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any coordinate systems can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of coordinate systems selected or null if menu cancelled

Example

To select coordinate systems from model m, flagging those selected with flag f, giving the prompt 'Select coordinate systems':

```
CoordinateSystem.Select(f, 'Select coordinate systems', m);
```

To select coordinate systems, flagging those selected with flag f but limiting selection to coordinate systems flagged with flag l, giving the prompt 'Select coordinate systems':

```
CoordinateSystem.Select(f, 'Select coordinate systems', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the coordinate system.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the coordinate system

Return type

No return value

Example

To set flag f for coordinate system c:

```
c.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the coordinate system. The coordinate system will be sketched until you either call CoordinateSystem.Unsketch(), CoordinateSystem.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the coordinate system is sketched. If omitted redraw is true. If you want to sketch several coordinate systems and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch coordinate system c:

```
c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged coordinate systems in the model. The coordinate systems will be sketched until you either call CoordinateSystem.Unsketch(), CoordinateSystem.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged coordinate systems will be sketched in
flag	Flag	Flag set on the coordinate systems that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the coordinate systems are sketched. If omitted redraw is true. If you want to sketch flagged coordinate systems several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all coordinate systems flagged with flag in model m:

```
CoordinateSystem.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing coordinate systems should be counted. If false or omitted referenced but undefined coordinate systems will also be included in the total.

Return type

number of coordinate systems

Example

To get the total number of coordinate systems in model m:

```
var total = CoordinateSystem.Total(m);
```

Unblank()

Description

Unblanks the coordinate system

Arguments

No arguments

Return type

No return value

Example

To unblank coordinate system c:

```
c.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the coordinate systems in the model.

Arguments

Name	Type	Description

Model	Model	Model that all coordinate systems will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the coordinate systems in model m:

```
CoordinateSystem.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged coordinate systems will be unblanked in
flag	Flag	Flag set on the coordinate systems that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the coordinate systems in model m flagged with f:

```
CoordinateSystem.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all coordinate systems will be unset in
flag	Flag	Flag to unset on the coordinate systems

Return type

No return value

Example

To unset the flag f on all the coordinate systems in model m:

```
CoordinateSystem.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the coordinate system.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the coordinate system is unsketched. If omitted redraw is true. If you want to unsketch several coordinate systems and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch coordinate system c:

```
c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all coordinate systems.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the coordinate systems are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all coordinate systems in model m:

```
CoordinateSystem.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged coordinate systems in the model.

Arguments

Name	Type	Description
Model	Model	Model that all coordinate systems will be unsketched in
flag	Flag	Flag set on the coordinate systems that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the coordinate systems are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all coordinate systems flagged with flag in model m:

```
CoordinateSystem.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

CoordinateSystem object.

Example

To check if CoordinateSystem property c.example is a parameter by using the CoordinateSystem.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for coordinate system. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for coordinate system c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this coordinate system.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for coordinate system c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the csys data in keyword format. Note that this contains the keyword header and the keyword cards. See also CoordinateSystem.Keyword() and CoordinateSystem.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for csys c in keyword format

```
var s = v.toString();
```

Curve class

The Curve class gives you access to load curve cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- CreateTable(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AddPoint(xvalue*[real]*, yvalue*[real]*)
- AddTableEntry(value*[real]*, load curve*[integer]*)
- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetPoint(row*[integer]*)
- GetTableEntry(row*[integer]*)
- InsertPoint(ipt*[integer]*, xvalue*[real]*, yvalue*[real]*, position*[integer]*)
- InsertTableEntry(ipt*[integer]*, value*[real]*, lcid*[integer]*, position*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemovePoint(row*[integer]*)
- RemoveTableEntry(ipt*[integer]*)
- SetFlag(flag*[Flag]*)
- SetPoint(ipt*[integer]*, xvalue*[real]*, yvalue*[real]*)
- SetTableEntry(ipt*[integer]*, value*[real]*, load curve*[integer]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Curve constants

Name	Description
Curve.AFTER	Insertion of curve data option.
Curve.BEFORE	Insertion of curve data option.
Curve.CURVE	Load curve type *DEFINE_CURVE
Curve.CURVE_FUNCTION	Load curve type *DEFINE_CURVE_FUNCTION
Curve.CURVE_SMOOTH	Load curve type *DEFINE_CURVE_SMOOTH
Curve.FUNCTION	Load curve type *DEFINE_FUNCTION
Curve.TABLE	Load curve type *DEFINE_TABLE

Curve properties

Name	Type	Description
dattyp	integer	Data type
dist	real	Total distance tool will travel
exists	logical	true if curve exists, false if referred to but not defined. (read only)
function	string	Function expression for Curve.CURVE_FUNCTION
heading	string	Curve heading
include	integer	The Include file number that the curve is in.
label	integer	Curve number. Also see the lcid property which is an alternative name for this.
lcid	integer	Curve number. Also see the label property which is an alternative name for this.
lcint	integer	Number of discretization points for the curve
model	integer	The Model number that the curve is in.
ncurves	integer	Number of points in curve or number of curves in table. The npoints property is an alternative name for this. (read only for tables)
npoints	integer	Number of points in curve or number of curves in table. The ncurves property is an alternative name for this. (read only for tables)
offa	real	Offset for abscissa values
offo	real	Offset for ordinate values
sfa	real	Scale factor on abscissa value
sfo	real	Scale factor on ordinate value
sidr	integer	Stress initialisation by dynamic relaxation
tend	real	Time curve returns to zero
trise	real	Rise time
tstart	real	Time curve starts to rise
type	constant	Load curve type (Can be Curve.CURVE, Curve.CURVE_FUNCTION, Curve.SMOOTH, Curve.FUNCTION or Curve.TABLE).
version	string	Version for discretization. Can be blank, "3858" or "5434a"
vmax	real	Maximum velocity

Detailed Description

The Curve class allows you to create, modify, edit and manipulate curve cards. See the documentation below for more details.

Constructor

new Curve(Load curve type*[constant]*, Model*[Model]*, lcid*[integer]*, sidr (optional)*[integer]*, sfa (optional)*[real]*, sfo (optional)*[real]*, offa (optional)*[real]*, offo (optional)*[real]*, dattyp (optional)*[integer]*, heading (optional)*[string]*, lcint (optional)*[integer]*)

Description

Create a new Curve object.

Arguments

Name	Type	Description
Load curve type	constant	Type of load curve. Can be Curve.CURVE, Curve.TABLE, Note this does not have to be defined. In previous versions of Primer you could only construct a basic load curve type, therefore the type argument was not used. Primer is still backwards compatible with this method of load curve creation.
Model	Model	Model that curve will be created in
lcid	integer	Curve number
sidr (optional)	integer	Stress initialisation by dynamic relaxation
sfa (optional)	real	Scale factor on abscissa value
sfo (optional)	real	Scale factor on ordinate value
offa (optional)	real	Offset on abscissa value
offo (optional)	real	Offset on ordinate value
dattyp (optional)	integer	Data type
heading (optional)	string	Title for the curve
lcint (optional)	integer	Number of discretization points for the curve

Return type

Curve object

Example

To create a new curve in model m with label 200

```
var l = new Curve(Curve.CURVE, m, 200);
```

new Curve(Load curve type*[constant]*, Model*[Model]*, lcid*[integer]*, sidr (optional)*[integer]*, function (optional)*[string]*, heading (optional)*[string]*)

Description

Create a new Curve *DEFINE_CURVE_FUNCTION object.

Arguments

Name	Type	Description
Load curve type	constant	Type of load curve. Must be Curve.CURVE_FUNCTION.
Model	Model	Model that curve will be created in
lcid	integer	Curve number
sidr (optional)	integer	Stress initialisation by dynamic relaxation
function (optional)	string	Function expression
heading (optional)	string	Title for the curve

Return type

Curve object

Example

To create a new curve function in model m with label 200 and function ’0.5*lc9*vm(22)**3’ (example from keyword manual)

```
var l = new Curve(Curve.CURVE_FUNCTION, m, 200, 0, "0.5*lc9*vm(22)**3");
```

new Curve(Load curve type*[constant]*, Model*[Model]*, lcid*[integer]*, function (optional)*[string]*, heading (optional)*[string]*)

Description

Create a new Curve *DEFINE_FUNCTION object.

Arguments

Name	Type	Description
Load curve type	constant	Type of load curve. Must be Curve.FUNCTION.
Model	Model	Model that curve will be created in
lcid	integer	Curve number
function (optional)	string	Function expression
heading (optional)	string	Title for the curve

Return type

Curve object

Example

To create a new function in model m with label 200 and function ’x(t)=1000*sin(100*t)’ and title ’x-velo’ (example from keyword manual)

```
var l = new Curve(Curve.FUNCTION, m, 200, "x(t)=1000*sin(100*t)", "x-velo");
```

new Curve(Load curve type*[constant]*, Model*[Model]*, lcid*[integer]*, sidr (optional)*[integer]*, dist (optional)*[real]*, tstart (optional)*[real]*, tend (optional)*[real]*, trise (optional)*[real]*, vmax (optional)*[real]*, heading (optional)*[string]*)

Description

Create a new Curve *DEFINE_CURVE_SMOOTH object.

Arguments

Name	Type	Description
Load curve type	constant	Type of load curve. Must be Curve.CURVE_SMOOTH.
Model	Model	Model that curve will be created in
lcid	integer	Curve number
sidr (optional)	integer	Stress initialisation by dynamic relaxation
dist (optional)	real	Total distance tool will travel
tstart (optional)	real	Time curve starts to rise
tend (optional)	real	Time curve returns to zero
trise (optional)	real	Rise time
vmax (optional)	real	Maximum velocity
heading (optional)	string	Title for the curve

Return type

Curve object

Example

To create a new curve function in model m with label 200’ (example from keyword manual)

```
var l = new Curve(Curve.CURVE_SMOOTH, m, 200);
```

Details of functions

AddPoint(xvalue*[real]*, yvalue*[real]*)

Description

Adds a point to a load curve.

Arguments

Name	Type	Description
xvalue	real	The x value of the point.
yvalue	real	The y value of the point.

Return type

No return value.

Example

To add a point with values of x=3 and y=5 to curve l:

```
l.AddPoint(3, 5);
```

AddTableEntry(value*[real]*, load curve*[integer]*)

Description

Adds an entry line to a table.

Arguments

Name	Type	Description
value	real	The value for for this entry in the table.
load curve	integer	The load curve corresponding to the defined value.

Return type

No return value.

Example

To add an entry with a value of 3 for load curve 1000:

```
l.AddTableEntry(3, 1000);
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a curve.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the curve

Return type

No return value

Example

To associate comment c to the curve c:

```
c.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse curve c:

```
c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the curve.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the curve

Return type

No return value

Example

To clear flag f for curve c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the curve.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Curve object

Example

To copy curve c into curve z:

```
var z = c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a curve.

Arguments

Name	Type	Description

Model	Model	Model that the curve will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Curve object (or null if not made)

Example

To start creating a curve in model m:

```
var l = Curve.Create(m);
```

CreateTable(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a table.

Arguments

Name	Type	Description
Model	Model	Model that the curve will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Curve object (or null if not made)

Example

To start creating a table in model m:

```
var l = Curve.CreateTable(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a curve.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the curve

Return type

No return value

Example

To detach comment c from the curve c:

```
c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit curve c:

```
c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for curve. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for curve c:

```
c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first curve in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first curve in

Return type

Curve object (or null if there are no curves in the model).

Example

To get the first curve in model m:

```
var c = Curve.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free curve label in the model. Also see Curve.LastFreeLabel(), Curve.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free curve label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Curve label.

Example

To get the first free curve label in model m:

```
var label = Curve.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the curves in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all curves will be flagged in
flag	Flag	Flag to set on the curves

Return type

No return value

Example

To flag all of the curves with flag f in model m:

```
Curve.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the curve is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the curve

Return type

true if flagged, false if not.

Example

To check if curve c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each curve in the model.
Note that ForEach has been designed to make looping over curves as fast as possible and so has some limitations. Firstly, a single temporary Curve object is created and on each function call it is updated with the current curve data. This means that you should not try to store the Curve object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new curves inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all curves are in
func	function	Function to call for each curve
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the curves in model m:

```
Curve.ForEach(m, test);
function test(c)
{
// c is Curve object
}
```

To call function test for all of the curves in model m with optional object:

```
var data = { x:0, y:0 };
Curve.ForEach(m, test, data);
function test(c, extra)
{
// c is Curve object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Curve objects for all of the curves in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get curves from

Return type

Array of Curve objects

Example

To make an array of Curve objects for all of the curves in model m

```
var c = Curve.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a curve.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the curve c:

```
var comm_array = c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Curve objects for all of the flagged curves in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get curves from
flag	Flag	Flag set on the curves that you want to retrieve

Return type

Array of Curve objects

Example

To make an array of Curve objects for all of the curves in model m flagged with f

```
var c = Curve.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Curve object for a curve ID.

Arguments

Name	Type	Description
Model	Model	Model to find the curve in
number	integer	number of the curve you want the Curve object for

Return type

Curve object (or null if curve does not exist).

Example

To get the Curve object for curve 100 in model m

```
var c = Curve.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Curve property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Curve.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	curve property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Curve property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Curve property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

GetPoint(row*[integer]*)

Description

Returns x and y data for a point in a curve

Arguments

Name	Type	Description
row	integer	The row point you want the data for. **Note that curve points start at 0, not 1**.

Return type

An array containing the x coordinate and the y coordinate.

Example

To get the curve data for the 3rd point for curve l:

```
if (l.npoints >= 3)
{
    var point_data = l.GetPoint(2);
}
```

GetTableEntry(row*[integer]*)

Description

Returns the value and curve label for a row in a table

Arguments

Name	Type	Description
row	integer	The row point you want the data for. **Note that curve points start at 0, not 1**.

Return type

An array containing the value and the load curve label.

Example

To get the data for the 3rd point for table t:

```
if (t.npoints >= 3)
{
    var row_data = t.GetTableEntry(2);
}
```

InsertPoint(ipt*[integer]*, xvalue*[real]*, yvalue*[real]*, position*[integer]*)

Description

Inserts point values before or after a specified row of data on a load curve.

Arguments

Name	Type	Description
ipt	integer	The row you want to insert the data before or after. **Note that the row data starts at 0, not 1**.
xvalue	real	The x value of the point.
yvalue	real	The y value of the point.
position	integer	Specify either before or after the selected row. Use ’Curve.BEFORE’ for before, and ’Curve.AFTER’ for after.

Return type

No return value.

Example

To insert the values after the 3rd row to x=3, y=5 for curve l:

```
l.InsertPoint(2, 3, 5, Curve.AFTER);
```

InsertTableEntry(ipt*[integer]*, value*[real]*, lcid*[integer]*, position*[integer]*)

Description

Inserts a table row before or after a specified row of data on a table.

Arguments

Name	Type	Description
ipt	integer	The row you want to insert the data before or after. **Note that the row data starts at 0, not 1**.
value	real	The value of the row.
lcid	integer	The load curve corresponding to the defined value.
position	integer	Specify either before or after the selected row. Use 'Curve.BEFORE' for before, and 'Curve.AFTER' for after.

Return type

No return value.

Example

To insert the values after the 3rd row to value=3, lcur=5 for table t:

```
t.InsertTableEntry(2, 3, 5, Curve.AFTER);
```

Keyword()

Description

Returns the keyword for this curve (*DEFINE_CURVE_xxxx). **Note that a carriage return is not added**. See also Curve.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for curve l:

```
var key = l.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the curve. **Note that a carriage return is not added**. See also Curve.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for curve l:

```
var cards = l.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last curve in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last curve in

Return type

Curve object (or null if there are no curves in the model).

Example

To get the last curve in model m:

```
var c = Curve.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free curve label in the model. Also see Curve.FirstFreeLabel(), Curve.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free curve label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Curve label.

Example

To get the last free curve label in model m:

```
var label = Curve.LastFreeLabel(m);
```

Next()

Description

Returns the next curve in the model.

Arguments

No arguments

Return type

Curve object (or null if there are no more curves in the model).

Example

To get the curve in model m after curve c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) curve label in the model. Also see Curve.FirstFreeLabel(), Curve.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free curve label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Curve label.

Example

To get the next free curve label in model m:

```
var label = Curve.NextFreeLabel(m);
```

Previous()

Description

Returns the previous curve in the model.

Arguments

No arguments

Return type

Curve object (or null if there are no more curves in the model).

Example

To get the curve in model m before curve c:

```
var c = c.Previous();
```

RemovePoint(row*[integer]*)

Description

Removes a row of data from a curve

Arguments

Name	Type	Description
row	integer	The row point you want to remove. **Note that curve points start at 0, not 1**.

Return type

No return value.

Example

To remove the curve data for the 3rd point for curve l:

```
if (l.npoints >= 3)
{
    var point_data = l.RemovePoint(2);
}
```

RemoveTableEntry(ipt*[integer]*)

Description

Removes the value and loadcurve values for a specified row of data on a load curve.

Arguments

Name	Type	Description
ipt	integer	The row you want to remove the data for. **Note that the row data starts at 0, not 1**.

Return type

No return value.

Example

To remove an entry at row 4:

```
t.RemoveTableEntry(4);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the curves in the model.

Arguments

Name	Type	Description
Model	Model	Model that all curves will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the curves in model m, from 1000000:

```
Curve.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged curves in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged curves will be renumbered in
flag	Flag	Flag set on the curves that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the curves in model m flagged with f, from 1000000:

```
Curve.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select curves using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting curves
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only curves from that model can be selected. If the argument is a Flag then only curves that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any curves can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of curves selected or null if menu cancelled

Example

To select curves from model m, flagging those selected with flag f, giving the prompt 'Select curves':

```
Curve.Select(f, 'Select curves', m);
```

To select curves, flagging those selected with flag f but limiting selection to curves flagged with flag l, giving the prompt 'Select curves':

```
Curve.Select(f, 'Select curves', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the curve.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the curve

Return type

No return value

Example

To set flag f for curve c:

```
c.SetFlag(f);
```

SetPoint(ipt*[integer]*, xvalue*[real]*, yvalue*[real]*)

Description

Sets the x and y values for a specified row of data on a load curve.

Arguments

Name	Type	Description
ipt	integer	The row you want to set the data for. **Note that the row data starts at 0, not 1**.
xvalue	real	The x value of the point.
yvalue	real	The y value of the point.

Return type

No return value.

Example

To set the values for the 3rd row to x=3, y=5 for curve l:

```
l.SetPoint(2, 3, 5);
```

SetTableEntry(ipt*[integer]*, value*[real]*, load curve*[integer]*)

Description

Sets the value and loadcurve values for a specified row of data on a load curve.

Arguments

Name	Type	Description
ipt	integer	The row you want to set the data for. **Note that the row data starts at 0, not 1**.
value	real	The value for for this entry in the table.
load curve	integer	The load curve corresponding to the defined value.

Return type

No return value.

Example

To add an entry with a value of 3 for load curve 1000 at row 4:

```
t.SetTableEntry(4, 3, 1000);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of curves in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing curves should be counted. If false or omitted referenced but undefined curves will also be included in the total.

Return type

number of curves

Example

To get the total number of curves in model m:

```
var total = Curve.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the curves in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all curves will be unset in
flag	Flag	Flag to unset on the curves

Return type

No return value

Example

To unset the flag f on all the curves in model m:

```
Curve.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Curve object.

Example

To check if Curve property c.example is a parameter by using the Curve.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for curve. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for curve c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this curve.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for curve c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the curve data in keyword format. Note that this contains the keyword header and the keyword cards. See also Curve.Keyword() and Curve.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for curve l in keyword format

```
var l = d.toString();
```

ElementDeath class

The ElementDeath class gives you access to define element death cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

ElementDeath constants

Name	Description
ElementDeath.BEAM	Beam option
ElementDeath.BEAM_SET	Beam set option
ElementDeath.SHELL	Shell option
ElementDeath.SHELL_SET	Shell set option
ElementDeath.SOLID	Solid option
ElementDeath.SOLID_SET	Solid set option

ElementDeath.THICK_SHELL	Thick shell option
ElementDeath.THICK_SHELL_SET	Thick shell set option

ElementDeath properties

Name	Type	Description
boxid	integer	Box restricting element deletion
cid	integer	Coordinate ID for transforming boxid.
eid	integer	Element ID or element set ID. The sid property is an alternative name for this.
exists	logical	true if element death exists, false if referred to but not defined. (read only)
idgrp	integer	Group ID for simultaneous deletion.
include	integer	The Include file number that the element death is in.
inout	logical	If true, LS_DYNA deletes elements outside box, otherwise inside box.
model	integer	The Model number that the element death is in.
option	constant	ElementDeath option. Can be ElementDeath.SOLID, ElementDeath.SOLID_SET, ElementDeath.BEAM, ElementDeath.BEAM_SET, ElementDeath.SHELL, ElementDeath.SHELL_SET, ElementDeath.THICK_SHELL or ElementDeath.THICK_SHELL_SET. The type property is an alternative name for this.
percent	real	Deletion percentage.
sid	integer	Element ID or element set ID. The eid property is an alternative name for this.
time	real	Deletion time for elimination
title	string	ElementDeath title
type	constant	ElementDeath option. Can be ElementDeath.SOLID, ElementDeath.SOLID_SET, ElementDeath.BEAM, ElementDeath.BEAM_SET, ElementDeath.SHELL, ElementDeath.SHELL_SET, ElementDeath.THICK_SHELL or ElementDeath.THICK_SHELL_SET The option property is an alternative name for this.

Detailed Description

The ElementDeath class allows you to create, modify, edit and manipulate element death cards. See the documentation below for more details.

Constructor

new ElementDeath(Model*[Model]*, type*[string]*, eid/sid*[integer]*)

Description

Create a new ElementDeath object.

Arguments

Name	Type	Description
Model	Model	Model that element death will be created in
type	string	ElementDeath type. Can be ElementDeath.SOLID, ElementDeath.SOLID_SET, ElementDeath.BEAM, ElementDeath.BEAM_SET, ElementDeath.SHELL, ElementDeath.SHELL_SET, ElementDeath.THICK_SHELL or ElementDeath.THICK_SHELL_SET
eid/sid	integer	Element or element set ID

Return type

ElementDeath object

Example

To create a new element death in model m with option BEAM_SET and sid 100

```
var ed = new ElementDeath(m, ElementDeath.BEAM_SET, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a element death.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the element death

Return type

No return value

Example

To associate comment c to the element death ed:

```
ed.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse element death ed:

```
ed.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the element death.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the element death

Return type

No return value

Example

To clear flag f for element death ed:

```
ed.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the element death.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

ElementDeath object

Example

To copy element death ed into element death z:

```
var z = ed.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an element death.

Arguments

Name	Type	Description
Model	Model	Model that the element death will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

ElementDeath object (or null if not made)

Example

To start creating an element death in model m:

```
var ed = ElementDeath.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a element death.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the element death

Return type

No return value

Example

To detach comment c from the element death ed:

```
ed.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit element death ed:

```
ed.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for element death. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for element death ed:

```
ed.Error("My custom error");
```

First(Model[*Model*]) [static]

Description

Returns the first element death in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first element death in

Return type

ElementDeath object (or null if there are no element deaths in the model).

Example

To get the first element death in model m:

```
var ed = ElementDeath.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the element deaths in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all element deaths will be flagged in
flag	Flag	Flag to set on the element deaths

Return type

No return value

Example

To flag all of the element deaths with flag f in model m:

```
ElementDeath.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the element death is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the element death

Return type

true if flagged, false if not.

Example

To check if element death ed has flag f set on it:

```
if (ed.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each element death in the model.
Note that ForEach has been designed to make looping over element deaths as fast as possible and so has some limitations.
Firstly, a single temporary ElementDeath object is created and on each function call it is updated with the current element death data. This means that you should not try to store the ElementDeath object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new element deaths inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all element deaths are in
func	function	Function to call for each element death
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the element deaths in model m:

```
ElementDeath.ForEach(m, test);
function test(ed)
{
// ed is ElementDeath object
}
```

To call function test for all of the element deaths in model m with optional object:

```
var data = { x:0, y:0 };
ElementDeath.ForEach(m, test, data);
function test(ed, extra)
{
// ed is ElementDeath object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of ElementDeath objects for all of the element deaths in a model in Primer

Arguments

Name	Type	Description

Model	Model	Model to get element deaths from

Return type

Array of ElementDeath objects

Example

To make an array of ElementDeath objects for all of the element deaths in model m

```
var ed = ElementDeath.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a element death.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the element death ed:

```
var comm_array = ed.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of ElementDeath objects for all of the flagged element deaths in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get element deaths from
flag	Flag	Flag set on the element deaths that you want to retrieve

Return type

Array of ElementDeath objects

Example

To make an array of ElementDeath objects for all of the element deaths in model m flagged with f

```
var ed = ElementDeath.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the ElementDeath object for a element death ID.

Arguments

Name	Type	Description
Model	Model	Model to find the element death in
number	integer	number of the element death you want the ElementDeath object for

Return type

ElementDeath object (or null if element death does not exist).

Example

To get the ElementDeath object for element death 100 in model m

```
var ed = ElementDeath.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a ElementDeath property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the ElementDeath.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	element death property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if ElementDeath property ed.example is a parameter:

```
Options.property_parameter_names = true;
if (ed.GetParameter(ed.example) ) do_something...
Options.property_parameter_names = false;
```

To check if ElementDeath property ed.example is a parameter by using the GetParameter method:

```
if (ed.ViewParameters().GetParameter(ed.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this element death (*DEFINE_ELEMENT_DEATH). **Note that a carriage return is not added**. See also ElementDeath.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for element death ed:

```
var key = ed.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the element death. **Note that a carriage return is not added**. See also ElementDeath.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for element death ed:

```
var cards = ed.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last element death in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last element death in

Return type

ElementDeath object (or null if there are no element deaths in the model).

Example

To get the last element death in model m:

```
var ed = ElementDeath.Last(m);
```

Next()

Description

Returns the next element death in the model.

Arguments

No arguments

Return type

ElementDeath object (or null if there are no more element deaths in the model).

Example

To get the element death in model m after element death ed:

```
var ed = ed.Next();
```

Previous()

Description

Returns the previous element death in the model.

Arguments

No arguments

Return type

ElementDeath object (or null if there are no more element deaths in the model).

Example

To get the element death in model m before element death ed:

```
var ed = ed.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select element deaths using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting element deaths
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only element deaths from that model can be selected. If the argument is a Flag then only element deaths that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any element deaths can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of element deaths selected or null if menu cancelled

Example

To select element deaths from model m, flagging those selected with flag f, giving the prompt 'Select element deaths':

```
ElementDeath.Select(f, 'Select element deaths', m);
```

To select element deaths, flagging those selected with flag f but limiting selection to element deaths flagged with flag l, giving the prompt 'Select element deaths':

```
ElementDeath.Select(f, 'Select element deaths', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the element death.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the element death

Return type

No return value

Example

To set flag f for element death ed:

```
ed.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of element deaths in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing element deaths should be counted. If false or omitted referenced but undefined element deaths will also be included in the total.

Return type

number of element deaths

Example

To get the total number of element deaths in model m:

```
var total = ElementDeath.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the element deaths in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all element deaths will be unset in
flag	Flag	Flag to unset on the element deaths

Return type

No return value

Example

To unset the flag f on all the element deaths in model m:

```
ElementDeath.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

ElementDeath object.

Example

To check if ElementDeath property ed.example is a parameter by using the ElementDeath.GetParameter() method:

```
if (ed.ViewParameters().GetParameter(ed.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for element death. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for element death ed:

```
ed.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this element death.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for element death ed:

```
var xrefs = ed.Xrefs();
```

---

toString()

Description

Creates a string containing the element death data in keyword format. Note that this contains the keyword header and the keyword cards. See also ElementDeath.Keyword() and ElementDeath.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for element death ed in keyword format

```
var s = ed.toString();
```

HexSpotweldAssembly class

The HexSpotweldAssembly class gives you access to *DEFINE_HEX_SPOTWELD_ASSEMBLY cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

HexSpotweldAssembly properties

Name	Type	Description
eid1	integer	EID 1
eid10	integer	EID 10
eid11	integer	EID 11

eid12	integer	EID 12
eid13	integer	EID 13
eid14	integer	EID 14
eid15	integer	EID 15
eid16	integer	EID 16
eid2	integer	EID 2
eid3	integer	EID 3
eid4	integer	EID 4
eid5	integer	EID 5
eid6	integer	EID 6
eid7	integer	EID 7
eid8	integer	EID 8
eid9	integer	EID 9
exists	logical	true if *DEFINE_HEX_SPOTWELD_ASSEMBLY exists, false if referred to but not defined. (read only)
id	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY id
include	integer	The Include file number that the *DEFINE_HEX_SPOTWELD_ASSEMBLY is in.
model	integer	The Model number that the DEFINE_HEX_SPOTWELD_ASSEMBLY is in.
opt	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY opt
title	string	Title (optional)

Detailed Description

The HexSpotweldAssembly class allows you to create, modify, edit and manipulate *DEFINE_HEX_SPOTWELD_ASSEMBLY cards. See the documentation below for more details.

Constructor

new HexSpotweldAssembly(Model*[Model]*, id*[integer]*, opt*[integer]*, eid1*[integer]*, eid2*[integer]*, eid3*[integer]*, eid4*[integer]*, title (optional)*[string]*)

Description

Create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY object.

Arguments

Name	Type	Description
Model	Model	Model that Hex Spotweld Assembly will be created in
id	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY id_sw.
opt	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY opt can be 4, 8 or 16
eid1	integer	EID 1.
eid2	integer	EID 2.
eid3	integer	EID 3.
eid4	integer	EID 4.

title (optional)	string	Define hex spotweld assembly title.

Return type

HexSpotweldAssembly object

Example

To create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY with ID 100 in model m with 4 elements 50, 150, 250 and 350

```
var h = new HexSpotweldAssembly(m, 100, 4, 50, 150, 250, 350);
```

new HexSpotweldAssembly(Model*[Model]*, id*[integer]*, opt*[integer]*, eid1*[integer]*, eid2*[integer]*, eid3*[integer]*, eid4*[integer]*, eid5*[integer]*, eid6*[integer]*, eid7*[integer]*, eid8*[integer]*, title (optional)*[string]*)

Description

Create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY object.

Arguments

Name	Type	Description
Model	Model	Model that Hex Spotweld Assembly will be created in
id	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY id.
opt	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY opt can be 4, 8 or 16
eid1	integer	EID 1.
eid2	integer	EID 2.
eid3	integer	EID 3.
eid4	integer	EID 4.
eid5	integer	EID 5.
eid6	integer	EID 6.
eid7	integer	EID 7.
eid8	integer	EID 8.
title (optional)	string	Define hex spotweld assembly title.

Return type

HexSpotweldAssembly object

Example

To create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY with ID 100 in model m with 8 elements 50, 150, 250, 350, 450, 550, 650 and 750

```
var h = new HexSpotweldAssembly(m, 100, 8, 50, 150, 250, 350, 450, 550, 650,
750);
```

new HexSpotweldAssembly(Model*[Model]*, id*[integer]*, opt*[integer]*, eid1*[integer]*, eid2*[integer]*, eid3*[integer]*, eid4*[integer]*, eid5*[integer]*, eid6*[integer]*, eid7*[integer]*, eid8*[integer]*, eid9*[integer]*, eid10*[integer]*, eid11*[integer]*, eid12*[integer]*, eid13*[integer]*, eid14*[integer]*, eid15*[integer]*, eid16*[integer]*, title (optional)*[string]*)

Description

Create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY object.

Arguments

Name	Type	Description
Model	Model	Model that Hex Spotweld Assembly will be created in
id	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY id.
opt	integer	*DEFINE_HEX_SPOTWELD_ASSEMBLY opt can be 4, 8 or 16
eid1	integer	EID 1.
eid2	integer	EID 2.
eid3	integer	EID 3.
eid4	integer	EID 4.
eid5	integer	EID 5.
eid6	integer	EID 6.
eid7	integer	EID 7.
eid8	integer	EID 8.
eid9	integer	EID 9.
eid10	integer	EID 10.
eid11	integer	EID 11.
eid12	integer	EID 12.
eid13	integer	EID 13.
eid14	integer	EID 14.
eid15	integer	EID 15.
eid16	integer	EID 16.
title (optional)	string	Define hex spotweld assembly title.

Return type

HexSpotweldAssembly object

Example

To create a new *DEFINE_HEX_SPOTWELD_ASSEMBLY with ID 100 in model m with 16 elements 50, 150, 250, 350, 450, 550, 650, 750, 850, 950, 1050, 1150, 1250, 1350, 1450 and 1550

```
var h = new HexSpotweldAssembly(m, 100, 16, 50, 150, 250, 350, 450, 550, 650,
750, 850, 950, 1050, 1150, 1250, 1350, 1450, 1550);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the DEFINE_HEX_SPOTWELD_ASSEMBLY

Return type

No return value

Example

To associate comment c to the DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the DEFINE_HEX_SPOTWELD_ASSEMBLY

Return type

No return value

Example

To clear flag f for DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

HexSpotweldAssembly object

Example

To copy DEFINE_HEX_SPOTWELD_ASSEMBLY h into DEFINE_HEX_SPOTWELD_ASSEMBLY z:

```
var z = h.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a HexSpotweldAssembly.

Arguments

Name	Type	Description
Model	Model	Model that the Hex Spotweld Assembly will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

HexSpotweldAssembly object (or null if not made)

Example

To start creating a HexSpotweldAssem in model s:

```
var s = HexSpotweldAssembly.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be detached from the DEFINE_HEX_SPOTWELD_ASSEMBLY

Return type

No return value

Example

To detach comment c from the DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for DEFINE_HEX_SPOTWELD_ASSEMBLY. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first DEFINE_HEX_SPOTWELD_ASSEMBLY in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first DEFINE_HEX_SPOTWELD_ASSEMBLY in

Return type

HexSpotweldAssembly object (or null if there are no DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model).

Example

To get the first DEFINE_HEX_SPOTWELD_ASSEMBLY in model m:

```
var h = HexSpotweldAssembly.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free DEFINE_HEX_SPOTWELD_ASSEMBLY label in the model. Also see HexSpotweldAssembly.LastFreeLabel(), HexSpotweldAssembly.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free DEFINE_HEX_SPOTWELD_ASSEMBLY label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

HexSpotweldAssembly label.

Example

To get the first free DEFINE_HEX_SPOTWELD_ASSEMBLY label in model m:

```
var label = HexSpotweldAssembly.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all DEFINE_HEX_SPOTWELD_ASSEMBLYs will be flagged in
flag	Flag	Flag to set on the DEFINE_HEX_SPOTWELD_ASSEMBLYs

Return type

No return value

Example

To flag all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs with flag f in model m:

```
HexSpotweldAssembly.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the DEFINE_HEX_SPOTWELD_ASSEMBLY is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the DEFINE_HEX_SPOTWELD_ASSEMBLY

Return type

true if flagged, false if not.

Example

To check if DEFINE_HEX_SPOTWELD_ASSEMBLY h has flag f set on it:

```
if (h.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each DEFINE_HEX_SPOTWELD_ASSEMBLY in the model.
Note that ForEach has been designed to make looping over DEFINE_HEX_SPOTWELD_ASSEMBLYs as fast as possible and so has some limitations.
Firstly, a single temporary HexSpotweldAssembly object is created and on each function call it is updated with the current DEFINE_HEX_SPOTWELD_ASSEMBLY data. This means that you should not try to store the HexSpotweldAssembly object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new DEFINE_HEX_SPOTWELD_ASSEMBLYs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all DEFINE_HEX_SPOTWELD_ASSEMBLYs are in
func	function	Function to call for each DEFINE_HEX_SPOTWELD_ASSEMBLY
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m:

```
HexSpotweldAssembly.ForEach(m, test);
function test(h)
{
// h is HexSpotweldAssembly object
}
```

To call function test for all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m with optional object:

```
var data = { x:0, y:0 };
HexSpotweldAssembly.ForEach(m, test, data);
function test(h, extra)
{
// h is HexSpotweldAssembly object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of HexSpotweldAssembly objects for all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get DEFINE_HEX_SPOTWELD_ASSEMBLYs from

Return type

Array of HexSpotweldAssembly objects

Example

To make an array of HexSpotweldAssembly objects for all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m

```
var h = HexSpotweldAssembly.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
var comm_array = h.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of HexSpotweldAssembly objects for all of the flagged DEFINE_HEX_SPOTWELD_ASSEMBLYs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get DEFINE_HEX_SPOTWELD_ASSEMBLYs from
flag	Flag	Flag set on the DEFINE_HEX_SPOTWELD_ASSEMBLYs that you want to retrieve

Return type

Array of HexSpotweldAssembly objects

Example

To make an array of HexSpotweldAssembly objects for all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m flagged with f

```
var h = HexSpotweldAssembly.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the HexSpotweldAssembly object for a DEFINE_HEX_SPOTWELD_ASSEMBLY ID.

Arguments

Name	Type	Description
Model	Model	Model to find the DEFINE_HEX_SPOTWELD_ASSEMBLY in
number	integer	number of the DEFINE_HEX_SPOTWELD_ASSEMBLY you want the HexSpotweldAssembly object for

Return type

HexSpotweldAssembly object (or null if DEFINE_HEX_SPOTWELD_ASSEMBLY does not exist).

Example

To get the HexSpotweldAssembly object for DEFINE_HEX_SPOTWELD_ASSEMBLY 100 in model m

```
var h = HexSpotweldAssembly.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a HexSpotweldAssembly property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the HexSpotweldAssembly.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	DEFINE_HEX_SPOTWELD_ASSEMBLY property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if HexSpotweldAssembly property h.example is a parameter:

```
Options.property_parameter_names = true;
if (h.GetParameter(h.example) ) do_something...
Options.property_parameter_names = false;
```

To check if HexSpotweldAssembly property h.example is a parameter by using the GetParameter method:

```
if (h.ViewParameters().GetParameter(h.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this HexSpotweldAssembly (*DEFINE_HEX_SPOTWELD_ASSEMBLY). **Note that a carriage return is not added**. See also HexSpotweldAssembly.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for HexSpotweldAssem s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the HexSpotweldAssem. **Note that a carriage return is not added**. See also HexSpotweldAssembly.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for HexSpotweldAssem s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last DEFINE_HEX_SPOTWELD_ASSEMBLY in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last DEFINE_HEX_SPOTWELD_ASSEMBLY in

Return type

HexSpotweldAssembly object (or null if there are no DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model).

Example

To get the last DEFINE_HEX_SPOTWELD_ASSEMBLY in model m:

```
var h = HexSpotweldAssembly.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free DEFINE_HEX_SPOTWELD_ASSEMBLY label in the model. Also see HexSpotweldAssembly.FirstFreeLabel(), HexSpotweldAssembly.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free DEFINE_HEX_SPOTWELD_ASSEMBLY label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

HexSpotweldAssembly label.

Example

To get the last free DEFINE_HEX_SPOTWELD_ASSEMBLY label in model m:

```
var label = HexSpotweldAssembly.LastFreeLabel(m);
```

Next()

Description

Returns the next DEFINE_HEX_SPOTWELD_ASSEMBLY in the model.

Arguments

No arguments

Return type

HexSpotweldAssembly object (or null if there are no more DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model).

Example

To get the DEFINE_HEX_SPOTWELD_ASSEMBLY in model m after DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
var h = h.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) DEFINE_HEX_SPOTWELD_ASSEMBLY label in the model. Also see HexSpotweldAssembly.FirstFreeLabel(), HexSpotweldAssembly.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free DEFINE_HEX_SPOTWELD_ASSEMBLY label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

HexSpotweldAssembly label.

Example

To get the next free DEFINE_HEX_SPOTWELD_ASSEMBLY label in model m:

```
var label = HexSpotweldAssembly.NextFreeLabel(m);
```

Previous()

Description

Returns the previous DEFINE_HEX_SPOTWELD_ASSEMBLY in the model.

Arguments

No arguments

Return type

HexSpotweldAssembly object (or null if there are no more DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model).

Example

To get the DEFINE_HEX_SPOTWELD_ASSEMBLY in model m before DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
var h = h.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all DEFINE_HEX_SPOTWELD_ASSEMBLYs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m, from 1000000:

```
HexSpotweldAssembly.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged DEFINE_HEX_SPOTWELD_ASSEMBLYs will be renumbered in
flag	Flag	Flag set on the DEFINE_HEX_SPOTWELD_ASSEMBLYs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m flagged with f, from 1000000:

```
HexSpotweldAssembly.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select DEFINE_HEX_SPOTWELD_ASSEMBLYs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting DEFINE_HEX_SPOTWELD_ASSEMBLYs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only DEFINE_HEX_SPOTWELD_ASSEMBLYs from that model can be selected. If the argument is a Flag then only DEFINE_HEX_SPOTWELD_ASSEMBLYs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any DEFINE_HEX_SPOTWELD_ASSEMBLYs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of DEFINE_HEX_SPOTWELD_ASSEMBLYs selected or null if menu cancelled

Example

To select DEFINE_HEX_SPOTWELD_ASSEMBLYs from model m, flagging those selected with flag f, giving the prompt 'Select DEFINE_HEX_SPOTWELD_ASSEMBLYs':

```
HexSpotweldAssembly.Select(f, 'Select DEFINE_HEX_SPOTWELD_ASSEMBLYs', m);
```

To select DEFINE_HEX_SPOTWELD_ASSEMBLYs, flagging those selected with flag f but limiting selection to DEFINE_HEX_SPOTWELD_ASSEMBLYs flagged with flag l, giving the prompt 'Select DEFINE_HEX_SPOTWELD_ASSEMBLYs':

```
HexSpotweldAssembly.Select(f, 'Select DEFINE_HEX_SPOTWELD_ASSEMBLYs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the DEFINE_HEX_SPOTWELD_ASSEMBLY

Return type

No return value

Example

To set flag f for DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing DEFINE_HEX_SPOTWELD_ASSEMBLYs should be counted. If false or omitted referenced but undefined DEFINE_HEX_SPOTWELD_ASSEMBLYs will also be included in the total.

Return type

number of DEFINE_HEX_SPOTWELD_ASSEMBLYs

Example

To get the total number of DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m:

```
var total = HexSpotweldAssembly.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the DEFINE_HEX_SPOTWELD_ASSEMBLYs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all DEFINE_HEX_SPOTWELD_ASSEMBLYs will be unset in
flag	Flag	Flag to unset on the DEFINE_HEX_SPOTWELD_ASSEMBLYs

Return type

No return value

Example

To unset the flag f on all the DEFINE_HEX_SPOTWELD_ASSEMBLYs in model m:

```
HexSpotweldAssembly.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

HexSpotweldAssembly object.

Example

To check if HexSpotweldAssembly property h.example is a parameter by using the HexSpotweldAssembly.GetParameter() method:

```
if (h.ViewParameters().GetParameter(h.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for DEFINE_HEX_SPOTWELD_ASSEMBLY. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
h.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this DEFINE_HEX_SPOTWELD_ASSEMBLY.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for DEFINE_HEX_SPOTWELD_ASSEMBLY h:

```
var xrefs = h.Xrefs();
```

---

toString()

Description

Creates a string containing the HexSpotweldAssem data in keyword format. Note that this contains the keyword header and the keyword cards. See also HexSpotweldAssembly.Keyword() and HexSpotweldAssembly.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for HexSpotweldAssem s in keyword format

```
var str = s.toString();
```

StagedConstructionPart class

The StagedConstructionPart class gives you access to Define staged construction part cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

StagedConstructionPart constants

Name	Description
StagedConstructionPart.PART	DEFN is *DEFINE_STAGED_CONSTRUCTION_PART.
StagedConstructionPart.SET	DEFN is *DEFINE_STAGED_CONSTRUCTION_PART_SET.

StagedConstructionPart properties

Name	Type	Description
exists	logical	true if Define staged construction parts exists, false if referred to but not defined (read only)
id	integer	Part ID or part set ID (not internal label)
include	integer	The Include file number that the Define staged construction parts is in.
label	integer	The label the Define staged construction parts has in PRIMER (read only)
model	integer	The Model number that the Define staged construction part is in.
option	constant	The Define staged construction parts option. Can be StagedConstructionPart.PART or StagedConstructionPart.SET.
stga	integer	Construction stage at which part is added.
stgr	integer	Construction stage at which part is removed.

Detailed Description

The StagedConstructionPart class allows you to create, modify, edit and manipulate Define staged construction parts cards. See the documentation below for more details.

Constructor

new StagedConstructionPart(Model*[Model]*, option*[constant]*, id*[integer]*, stga*[integer]*, stgr*[integer]*)

Description

Create a new StagedConstructionPart object.

Arguments

Name	Type	Description
Model	Model	Model that Define staged construction parts will be created in
option	constant	Specify the type of Define staged construction parts. Can be StagedConstructionPart.PART or StagedConstructionPart.SET)
id	integer	Part ID or part set ID
stga	integer	Construction stage at which part is added.
stgr	integer	Construction stage at which part is removed.

Return type

StagedConstructionPart object

Example

To create a new Define staged construction part in model m, of type SET, with part set 9, stga 18 and stgr 12

```
var scp = new StagedConstructionPart(m, StagedConstructionPart.SET, 9, 18, 12);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a Define staged construction part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the Define staged construction part

Return type

No return value

Example

To associate comment c to the Define staged construction part scp:

```
scp.AssociateComment(c);
```

Blank()

Description

Blanks the Define staged construction part

Arguments

No arguments

Return type

No return value

Example

To blank Define staged construction part scp:

```
scp.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the Define staged construction parts in model m:

```
StagedConstructionPart.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged Define staged construction parts will be blanked in
flag	Flag	Flag set on the Define staged construction parts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the Define staged construction parts in model m flagged with f:

```
StagedConstructionPart.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the Define staged construction part is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if Define staged construction part scp is blanked:

```
if (scp.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse Define staged construction part scp:

```
scp.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the Define staged construction part.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the Define staged construction part

Return type

No return value

Example

To clear flag f for Define staged construction part scp:

```
scp.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the Define staged construction part.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

StagedConstructionPart object

Example

To copy Define staged construction part scp into Define staged construction part z:

```
var z = scp.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a Define staged construction parts card.

Arguments

Name	Type	Description
Model	Model	Model that the Define staged construction parts card will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

StagedConstructionPart object (or null if not made)

Example

To start creating a Define staged construction parts card in model m:

```
var scp = StagedConstructionPart.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a Define staged construction part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the Define staged construction part

Return type

No return value

Example

To detach comment c from the Define staged construction part scp:

```
scp.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit Define staged construction part scp:

```
scp.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for Define staged construction part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for Define staged construction part scp:

```
scp.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first Define staged construction part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first Define staged construction part in

Return type

StagedConstructionPart object (or null if there are no Define staged construction parts in the model).

Example

To get the first Define staged construction part in model m:

```
var scp = StagedConstructionPart.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the Define staged construction parts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts will be flagged in
flag	Flag	Flag to set on the Define staged construction parts

Return type

No return value

Example

To flag all of the Define staged construction parts with flag f in model m:

```
StagedConstructionPart.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the Define staged construction part is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the Define staged construction part

Return type

true if flagged, false if not.

Example

To check if Define staged construction part scp has flag f set on it:

```
if (scp.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each Define staged construction part in the model.
Note that ForEach has been designed to make looping over Define staged construction parts as fast as possible and so has some limitations.
Firstly, a single temporary StagedConstructionPart object is created and on each function call it is updated with the current Define staged construction part data. This means that you should not try to store the StagedConstructionPart object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new Define staged construction parts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts are in
func	function	Function to call for each Define staged construction part
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the Define staged construction parts in model m:

```
StagedConstructionPart.ForEach(m, test);
function test(scp)
{
// scp is StagedConstructionPart object
}
```

To call function test for all of the Define staged construction parts in model m with optional object:

```
var data = { x:0, y:0 };
StagedConstructionPart.ForEach(m, test, data);
function test(scp, extra)
{
// scp is StagedConstructionPart object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of StagedConstructionPart objects for all of the Define staged construction parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get Define staged construction parts from

Return type

Array of StagedConstructionPart objects

Example

To make an array of StagedConstructionPart objects for all of the Define staged construction parts in model m

```
var scp = StagedConstructionPart.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a Define staged construction part.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the Define staged construction part scp:

```
var comm_array = scp.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of StagedConstructionPart objects for all of the flagged Define staged construction parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get Define staged construction parts from
flag	Flag	Flag set on the Define staged construction parts that you want to retrieve

Return type

Array of StagedConstructionPart objects

Example

To make an array of StagedConstructionPart objects for all of the Define staged construction parts in model m flagged with f

```
var scp = StagedConstructionPart.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the StagedConstructionPart object for a Define staged construction part ID.

Arguments

Name	Type	Description
Model	Model	Model to find the Define staged construction part in
number	integer	number of the Define staged construction part you want the StagedConstructionPart object for

Return type

StagedConstructionPart object (or null if Define staged construction part does not exist).

Example

To get the StagedConstructionPart object for Define staged construction part 100 in model m

```
var scp = StagedConstructionPart.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a StagedConstructionPart property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the StagedConstructionPart.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	Define staged construction part property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if StagedConstructionPart property scp.example is a parameter:

```
Options.property_parameter_names = true;
if (scp.GetParameter(scp.example) ) do_something...
Options.property_parameter_names = false;
```

To check if StagedConstructionPart property scp.example is a parameter by using the GetParameter method:

```
if (scp.ViewParameters().GetParameter(scp.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this Define staged construction parts (*Define_staged_construction_part). **Note that a carriage return is not added**. See also StagedConstructionPart.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Define staged construction parts scp:

```
var key = scp.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Define staged construction parts. **Note that a carriage return is not added**. See also StagedConstructionPart.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Define staged construction parts scp:

```
var cards = scp.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last Define staged construction part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last Define staged construction part in

Return type

StagedConstructionPart object (or null if there are no Define staged construction parts in the model).

Example

To get the last Define staged construction part in model m:

```
var scp = StagedConstructionPart.Last(m);
```

Next()

Description

Returns the next Define staged construction part in the model.

Arguments

No arguments

Return type

StagedConstructionPart object (or null if there are no more Define staged construction parts in the model).

Example

To get the Define staged construction part in model m after Define staged construction part scp:

```
var scp = scp.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a Define staged construction part.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only Define staged construction parts from that model can be picked. If the argument is a Flag then only Define staged construction parts that are flagged with *limit* can be selected. If omitted, or null, any Define staged construction parts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.

button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

StagedConstructionPart object (or null if not picked)

Example

To pick a Define staged construction part from model m giving the prompt 'Pick Define staged construction part from screen':

```
var scp = StagedConstructionPart.Pick('Pick Define staged construction part from screen', m);
```

Previous()

Description

Returns the previous Define staged construction part in the model.

Arguments

No arguments

Return type

StagedConstructionPart object (or null if there are no more Define staged construction parts in the model).

Example

To get the Define staged construction part in model m before Define staged construction part scp:

```
var scp = scp.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select Define staged construction parts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting Define staged construction parts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only Define staged construction parts from that model can be selected. If the argument is a Flag then only Define staged construction parts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any Define staged construction parts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of Define staged construction parts selected or null if menu cancelled

Example

To select Define staged construction parts from model m, flagging those selected with flag f, giving the prompt 'Select Define staged construction parts':

```
StagedConstructionPart.Select(f, 'Select Define staged construction parts', m);
```

To select Define staged construction parts, flagging those selected with flag f but limiting selection to Define staged construction parts flagged with flag l, giving the prompt 'Select Define staged construction parts':

```
StagedConstructionPart.Select(f, 'Select Define staged construction parts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the Define staged construction part.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the Define staged construction part

Return type

No return value

Example

To set flag f for Define staged construction part scp:

```
scp.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the Define staged construction part. The Define staged construction part will be sketched until you either call StagedConstructionPart.Unsketch(), StagedConstructionPart.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the Define staged construction part is sketched. If omitted redraw is true. If you want to sketch several Define staged construction parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch Define staged construction part scp:

```
scp.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged Define staged construction parts in the model. The Define staged construction parts will be sketched until you either call StagedConstructionPart.Unsketch(), StagedConstructionPart.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged Define staged construction parts will be sketched in
flag	Flag	Flag set on the Define staged construction parts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the Define staged construction parts are sketched. If omitted redraw is true. If you want to sketch flagged Define staged construction parts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all Define staged construction parts flagged with flag in model m:

```
StagedConstructionPart.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing Define staged construction parts should be counted. If false or omitted referenced but undefined Define staged construction parts will also be included in the total.

Return type

number of Define staged construction parts

Example

To get the total number of Define staged construction parts in model m:

```
var total = StagedConstructionPart.Total(m);
```

Unblank()

Description

Unblanks the Define staged construction part

Arguments

No arguments

Return type

No return value

Example

To unblank Define staged construction part scp:

```
scp.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the Define staged construction parts in model m:

```
StagedConstructionPart.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged Define staged construction parts will be unblanked in
flag	Flag	Flag set on the Define staged construction parts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the Define staged construction parts in model m flagged with f:

```
StagedConstructionPart.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all Define staged construction parts will be unset in

flag	Flag	Flag to unset on the Define staged construction parts

Return type

No return value

Example

To unset the flag f on all the Define staged construction parts in model m:

```
StagedConstructionPart.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the Define staged construction part.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the Define staged construction part is unsketched. If omitted redraw is true. If you want to unsketch several Define staged construction parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch Define staged construction part scp:

```
scp.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all Define staged construction parts.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the Define staged construction parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all Define staged construction parts in model m:

```
StagedConstructionPart.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged Define staged construction parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all Define staged construction parts will be unsketched in
flag	Flag	Flag set on the Define staged construction parts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the Define staged construction parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all Define staged construction parts flagged with flag in model m:

```
StagedConstructionPart.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

StagedConstructionPart object.

Example

To check if StagedConstructionPart property scp.example is a parameter by using the StagedConstructionPart.GetParameter() method:

```
if (scp.ViewParameters().GetParameter(scp.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for Define staged construction part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give

details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for Define staged construction part scp:

```
scp.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this Define staged construction part.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for Define staged construction part scp:

```
var xrefs = scp.Xrefs();
```

---

toString()

Description

Creates a string containing the Define staged construction parts data in keyword format. Note that this contains the keyword header and the keyword cards. See also StagedConstructionPart.Keyword() and StagedConstructionPart.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Define staged construction parts scp in keyword format

```
var s = scp.toString();
```

Transformation class

The Transformation class gives you access to define transform cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AddRow(data*[Array of data]*, row (optional)*[integer]*)
- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRow(row*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRow(row*[integer]*)
- SetFlag(flag*[Flag]*)
- SetRow(row*[integer]*, data*[Array of data]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Transformation properties

Name	Type	Description
exists	logical	true if transformation exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the transformation is in.
label	integer	Transformation number. Also see the tranid property which is an alternative name for this.

model	integer	The Model number that the transformation is in.
nrow (read only)	integer	Number of rows of transformations
title	string	The title for the transformation.
tranid	integer	Transformation number. Also see the label property which is an alternative name for this.

Detailed Description

The Transformation class allows you to create, modify, edit and manipulate define transformation cards. See the documentation below for more details.

Constructor

new Transformation(Model*[Model]*, tranid*[integer]*, title (optional)*[string]*)

Description

Create a new Transformation object.

Arguments

Name	Type	Description
Model	Model	Model that transformation will be created in
tranid	integer	Transformation label
title (optional)	string	Transformation title

Return type

Transformation object

Example

To create a new transformation in model m with label 1000 and title "Example transform"

```
var t = new Transformation(m, 1000, "Example transform");
```

Details of functions

AddRow(data*[Array of data]*, row (optional)*[integer]*)

Description

Adds a row of data for a *DEFINE_TRANSFORMATION.

Arguments

Name	Type	Description
data	Array of data	The data you want to add
row (optional)	integer	The row you want to add the data at. Existing transforms will be shifted. If omitted the data will be added to the end of the existing transforms. **Note that row indices start at 0**.

Return type

No return value.

Example

To add a translation of (0, 0, 100) to transformation t:

```
var array = ["TRANSL", 0, 0, 100];
t.AddRow(array);
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a transformation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the transformation

Return type

No return value

Example

To associate comment c to the transformation t:

```
t.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse transformation t:

```
t.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the transformation.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the transformation

Return type

No return value

Example

To clear flag f for transformation t:

```
t.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the transformation.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Transformation object

Example

To copy transformation t into transformation z:

```
var z = t.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a define transformation definition.

Arguments

Name	Type	Description
Model	Model	Model that the transformation will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Transformation object (or null if not made)

Example

To start creating a define transformation definition in model m:

```
var t = Transformation.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a transformation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the transformation

Return type

No return value

Example

To detach comment c from the transformation t:

```
t.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit transformation t:

```
t.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for transformation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for transformation t:

```
t.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first transformation in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first transformation in

Return type

Transformation object (or null if there are no transformations in the model).

Example

To get the first transformation in model m:

```
var t = Transformation.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free transformation label in the model. Also see Transformation.LastFreeLabel(), Transformation.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free transformation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Transformation label.

Example

To get the first free transformation label in model m:

```
var label = Transformation.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the transformations in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all transformations will be flagged in
flag	Flag	Flag to set on the transformations

Return type

No return value

Example

To flag all of the transformations with flag f in model m:

```
Transformation.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the transformation is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the transformation

Return type

true if flagged, false if not.

Example

To check if transformation t has flag f set on it:

```
if (t.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each transformation in the model.
Note that ForEach has been designed to make looping over transformations as fast as possible and so has some limitations.
Firstly, a single temporary Transformation object is created and on each function call it is updated with the current transformation data. This means that you should not try to store the Transformation object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new transformations inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all transformations are in
func	function	Function to call for each transformation
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the transformations in model m:

```
Transformation.ForEach(m, test);
function test(t)
{
// t is Transformation object
}
```

To call function test for all of the transformations in model m with optional object:

```
var data = { x:0, y:0 };
Transformation.ForEach(m, test, data);
function test(t, extra)
{
// t is Transformation object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Transformation objects for all of the transformations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get transformations from

Return type

Array of Transformation objects

Example

To make an array of Transformation objects for all of the transformations in model m

```
var t = Transformation.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a transformation.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the transformation t:

```
var comm_array = t.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of Transformation objects for all of the flagged transformations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get transformations from
flag	Flag	Flag set on the transformations that you want to retrieve

Return type

Array of Transformation objects

Example

To make an array of Transformation objects for all of the transformations in model m flagged with f

```
var t = Transformation.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Transformation object for a transformation ID.

Arguments

Name	Type	Description
Model	Model	Model to find the transformation in
number	integer	number of the transformation you want the Transformation object for

Return type

Transformation object (or null if transformation does not exist).

Example

To get the Transformation object for transformation 100 in model m

```
var t = Transformation.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Transformation property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Transformation.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	transformation property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Transformation property t.example is a parameter:

```
Options.property_parameter_names = true;
if (t.GetParameter(t.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Transformation property t.example is a parameter by using the GetParameter method:

```
if (t.ViewParameters().GetParameter(t.example) ) do_something...
```

GetRow(row*[integer]*)

Description

Returns the data for a row in the transformation.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

An array of numbers containing the row variables.

Example

To get the data for the 2nd row in transformation t:

```
var data = t.GetRow(1);
```

Keyword()

Description

Returns the keyword for this transformation. **Note that a carriage return is not added**. See also Transformation.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for transformation t:

```
var key = t.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the transformation. **Note that a carriage return is not added**. See also Transformation.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for transformation i:

```
var cards = i.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last transformation in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last transformation in

Return type

Transformation object (or null if there are no transformations in the model).

Example

To get the last transformation in model m:

```
var t = Transformation.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free transformation label in the model. Also see Transformation.FirstFreeLabel(), Transformation.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free transformation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Transformation label.

Example

To get the last free transformation label in model m:

```
var label = Transformation.LastFreeLabel(m);
```

Next()

Description

Returns the next transformation in the model.

Arguments

No arguments

Return type

Transformation object (or null if there are no more transformations in the model).

Example

To get the transformation in model m after transformation t:

```
var t = t.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) transformation label in the model. Also see Transformation.FirstFreeLabel(), Transformation.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free transformation label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Transformation label.

Example

To get the next free transformation label in model m:

```
var label = Transformation.NextFreeLabel(m);
```

Previous()

Description

Returns the previous transformation in the model.

Arguments

No arguments

Return type

Transformation object (or null if there are no more transformations in the model).

Example

To get the transformation in model m before transformation t:

```
var t = t.Previous();
```

RemoveRow(row*[integer]*)

Description

Removes the data for a row in *DEFINE_TRANSFORMATION.

Arguments

Name	Type	Description
row	integer	The row you want to remove the data for. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second row of data for transformation t:

```
t.RemoveRow(1);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select transformations using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting transformations
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only transformations from that model can be selected. If the argument is a Flag then only transformations that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any transformations can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of transformations selected or null if menu cancelled

Example

To select transformations from model m, flagging those selected with flag f, giving the prompt 'Select transformations':

```
Transformation.Select(f, 'Select transformations', m);
```

To select transformations, flagging those selected with flag f but limiting selection to transformations flagged with flag l, giving the prompt 'Select transformations':

```
Transformation.Select(f, 'Select transformations', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the transformation.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the transformation

Return type

No return value

Example

To set flag f for transformation t:

```
t.SetFlag(f);
```

SetRow(row*[integer]*, data*[Array of data]*)

Description

Sets the data for a row in *DEFINE_TRANSFORMATION.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that row indices start at 0**.
data	Array of data	The data you want to set the row to

Return type

No return value.

Example

To set the second row of data for transformation t to be a translation of (0, 0, 100):

```
var array = ["TRANSL", 0, 0, 100];
t.SetRow(1, array);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of transformations in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing transformations should be counted. If false or omitted referenced but undefined transformations will also be included in the total.

Return type

number of transformations

Example

To get the total number of transformations in model m:

```
var total = Transformation.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the transformations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all transformations will be unset in
flag	Flag	Flag to unset on the transformations

Return type

No return value

Example

To unset the flag f on all the transformations in model m:

```
Transformation.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Transformation object.

Example

To check if Transformation property t.example is a parameter by using the Transformation.GetParameter() method:

```
if (t.ViewParameters().GetParameter(t.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for transformation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for transformation t:

```
t.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this transformation.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for transformation t:

```
var xrefs = t.Xrefs();
```

---

toString()

Description

Creates a string containing the transformation data in keyword format. Note that this contains the keyword header and the keyword cards. See also Transformation.Keyword() and Transformation.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for transformation t in keyword format

```
var s = t.toString();
```

Vector class

The Vector class gives you access to define vector cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Vector properties

Name	Type	Description
cid	int	Coordinate system ID
exists	logical	true if vector exists, false if referred to but not defined. (read only)
heading	string	Vector heading
include	integer	The Include file number that the vector is in.
label	integer	Vector number. Also see the vid property which is an alternative name for this.
model	integer	The Model number that the vector is in.
nodeh	int	Node ID for head of vector (for _NODES option)
nodes	logical	_NODES option
nodet	int	Node ID for tail of vector (for _NODES option)
vid	integer	Vector number. Also see the label property which is an alternative name for this.
xh	real	X coordinate of head of vector
xt	real	X coordinate of tail of vector
yh	real	Y coordinate of head of vector
yt	real	Y coordinate of tail of vector
zh	real	Z coordinate of head of vector
zt	real	Z coordinate of tail vector

Detailed Description

The Vector class allows you to create, modify, edit and manipulate vector cards. See the documentation below for more details.

Constructor

new Vector(Model*[Model]*, vid*[integer]*, xt*[real]*, yt*[real]*, zt*[real]*, xh*[real]*, yh*[real]*, zh*[real]*, cid (optional)*[int]*, heading (optional)*[string]*)

Description

Create a new Vector object.

Arguments

Name	Type	Description
Model	Model	Model that vector will be created in
vid	integer	Vector number
xt	real	X coordinate of tail of vector
yt	real	Y coordinate of tail of vector
zt	real	Z coordinate of tail vector
xh	real	X coordinate of head of vector
yh	real	Y coordinate of head of vector
zh	real	Z coordinate of head of vector
cid (optional)	int	Coordinate system ID

heading (optional)	string	Title for the vector

Return type

Vector object

Example

To create a new vector in model m with label 200

```
var v = new Vector(m, 200, 1.5, 2.5, 1.0, 4.5, 4.0, 3.0);
```

new Vector(Model*[Model]*, vid*[integer]*, nodet*[integer]*, nodeh*[integer]*, heading (optional)*[string]*)

Description

Create a new Vector object with _NODES option.

Arguments

Name	Type	Description
Model	Model	Model that vector will be created in
vid	integer	Vector number
nodet	integer	Node ID for tail of vector
nodeh	integer	Node ID for head of vector
heading (optional)	string	Title for the vector

Return type

Vector object

Example

To create a new vector in model m with label 200 using nodes 10 for the tail and 20 for the head

```
var v = new Vector(m, 200, 20, 30);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a vector.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the vector

Return type

No return value

Example

To associate comment c to the vector v:

```
v.AssociateComment(c);
```

Blank()

Description

Blanks the vector

Arguments

No arguments

Return type

No return value

Example

To blank vector v:

```
v.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the vectors in model m:

```
Vector.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged vectors will be blanked in
flag	Flag	Flag set on the vectors that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the vectors in model m flagged with f:

```
Vector.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the vector is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if vector v is blanked:

```
if (v.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse vector v:

```
v.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the vector.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the vector

Return type

No return value

Example

To clear flag f for vector v:

```
v.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the vector.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Vector object

Example

To copy vector v into vector z:

```
var z = v.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a vector.

Arguments

Name	Type	Description
Model	Model	Model that the vector will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Vector object (or null if not made)

Example

To start creating a vector in model m:

```
var m = Vector.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a vector.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the vector

Return type

No return value

Example

To detach comment c from the vector v:

```
v.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit vector v:

```
v.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for vector. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for vector v:

```
v.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first vector in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first vector in

Return type

Vector object (or null if there are no vectors in the model).

Example

To get the first vector in model m:

```
var v = Vector.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free vector label in the model. Also see Vector.LastFreeLabel(), Vector.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free vector label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Vector label.

Example

To get the first free vector label in model m:

```
var label = Vector.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the vectors in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be flagged in
flag	Flag	Flag to set on the vectors

Return type

No return value

Example

To flag all of the vectors with flag f in model m:

```
Vector.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the vector is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the vector

Return type

true if flagged, false if not.

Example

To check if vector v has flag f set on it:

```
if (v.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each vector in the model.
Note that ForEach has been designed to make looping over vectors as fast as possible and so has some limitations.
Firstly, a single temporary Vector object is created and on each function call it is updated with the current vector data. This means that you should not try to store the Vector object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new vectors inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all vectors are in
func	function	Function to call for each vector
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the vectors in model m:

```
Vector.ForEach(m, test);
function test(v)
{
// v is Vector object
}
```

To call function test for all of the vectors in model m with optional object:

```
var data = { x:0, y:0 };
Vector.ForEach(m, test, data);
function test(v, extra)
{
// v is Vector object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Vector objects for all of the vectors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get vectors from

Return type

Array of Vector objects

Example

To make an array of Vector objects for all of the vectors in model m

```
var v = Vector.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a vector.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the vector v:

```
var comm_array = v.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of Vector objects for all of the flagged vectors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get vectors from
flag	Flag	Flag set on the vectors that you want to retrieve

Return type

Array of Vector objects

Example

To make an array of Vector objects for all of the vectors in model m flagged with f

```
var v = Vector.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Vector object for a vector ID.

Arguments

Name	Type	Description
Model	Model	Model to find the vector in
number	integer	number of the vector you want the Vector object for

Return type

Vector object (or null if vector does not exist).

Example

To get the Vector object for vector 100 in model m

```
var v = Vector.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Vector property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Vector.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	vector property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Vector property v.example is a parameter:

```
Options.property_parameter_names = true;
if (v.GetParameter(v.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Vector property v.example is a parameter by using the GetParameter method:

```
if (v.ViewParameters().GetParameter(v.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this vector (*DEFINE_VECTOR). **Note that a carriage return is not added**. See also Vector.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for vector m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the vector. **Note that a carriage return is not added**. See also Vector.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for vector v:

```
var cards = v.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last vector in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last vector in

Return type

Vector object (or null if there are no vectors in the model).

Example

To get the last vector in model m:

```
var v = Vector.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free vector label in the model. Also see Vector.FirstFreeLabel(), Vector.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free vector label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Vector label.

Example

To get the last free vector label in model m:

```
var label = Vector.LastFreeLabel(m);
```

Next()

Description

Returns the next vector in the model.

Arguments

No arguments

Return type

Vector object (or null if there are no more vectors in the model).

Example

To get the vector in model m after vector v:

```
var v = v.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) vector label in the model. Also see Vector.FirstFreeLabel(), Vector.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free vector label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Vector label.

Example

To get the next free vector label in model m:

```
var label = Vector.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a vector.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only vectors from that model can be picked. If the argument is a Flag then only vectors that are flagged with *limit* can be selected. If omitted, or null, any vectors from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Vector object (or null if not picked)

Example

To pick a vector from model m giving the prompt 'Pick vector from screen':

```
var v = Vector.Pick('Pick vector from screen', m);
```

Previous()

Description

Returns the previous vector in the model.

Arguments

No arguments

Return type

Vector object (or null if there are no more vectors in the model).

Example

To get the vector in model m before vector v:

```
var v = v.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be renumbered in

start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the vectors in model m, from 1000000:

```
Vector.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged vectors will be renumbered in
flag	Flag	Flag set on the vectors that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the vectors in model m flagged with f, from 1000000:

```
Vector.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select vectors using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting vectors
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only vectors from that model can be selected. If the argument is a Flag then only vectors that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any vectors can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of vectors selected or null if menu cancelled

Example

To select vectors from model m, flagging those selected with flag f, giving the prompt 'Select vectors':

```
Vector.Select(f, 'Select vectors', m);
```

To select vectors, flagging those selected with flag f but limiting selection to vectors flagged with flag l, giving the prompt 'Select vectors':

```
Vector.Select(f, 'Select vectors', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the vector.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the vector

Return type

No return value

Example

To set flag f for vector v:

```
v.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the vector. The vector will be sketched until you either call Vector.Unsketch(), Vector.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the vector is sketched. If omitted redraw is true. If you want to sketch several vectors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch vector v:

```
v.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged vectors in the model. The vectors will be sketched until you either call Vector.Unsketch(), Vector.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged vectors will be sketched in
flag	Flag	Flag set on the vectors that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the vectors are sketched. If omitted redraw is true. If you want to sketch flagged vectors several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all vectors flagged with flag in model m:

```
Vector.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing vectors should be counted. If false or omitted referenced but undefined vectors will also be included in the total.

Return type

number of vectors

Example

To get the total number of vectors in model m:

```
var total = Vector.Total(m);
```

Unblank()

Description

Unblanks the vector

Arguments

No arguments

Return type

No return value

Example

To unblank vector v:

```
v.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the vectors in model m:

```
Vector.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged vectors will be unblanked in
flag	Flag	Flag set on the vectors that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the vectors in model m flagged with f:

```
Vector.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all vectors will be unset in

flag	Flag	Flag to unset on the vectors

Return type

No return value

Example

To unset the flag f on all the vectors in model m:

```
Vector.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the vector.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the vector is unsketched. If omitted redraw is true. If you want to unsketch several vectors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch vector v:

```
v.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all vectors.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the vectors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all vectors in model m:

```
Vector.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged vectors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all vectors will be unsketched in
flag	Flag	Flag set on the vectors that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the vectors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all vectors flagged with flag in model m:

```
Vector.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Vector object.

Example

To check if Vector property v.example is a parameter by using the Vector.GetParameter() method:

```
if (v.ViewParameters().GetParameter(v.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for vector. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for vector v:

```
v.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this vector.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for vector v:

```
var xrefs = v.Xrefs();
```

---

toString()

Description

Creates a string containing the vector data in keyword format. Note that this contains the keyword header and the keyword cards. See also Vector.Keyword() and Vector.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for vector v in keyword format

```
var s = v.toString();
```

DeformableToRigid class

The DeformableToRigid class gives you access to *DEFORMABLE_TO_RIGID cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetDefToRegAutoCard(ctype*[integer]*, index*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveDefToRegAutoCard(ctype*[integer]*, index*[integer]*)
- SetDefToRegAutoCard(ctype*[integer]*, index*[integer]*, ptype*[integer]*, pid*[integer]*, lrb (optional)*[integer]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()

- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

DeformableToRigid constants

Constants for DEFORMABLE_TO_RIGID type

Name	Description
DeformableToRigid.AUTOMATIC	*DEFORMABLE_TO_RIGID_AUTOMATIC.
DeformableToRigid.INERTIA	*DEFORMABLE_TO_RIGID_INERTIA.
DeformableToRigid.SIMPLE	*DEFORMABLE_TO_RIGID.

Constants for PID field type

Name	Description
DeformableToRigid.PART	Identifies the PID type as Part. Used for field ptype. Used only for DeformableToRigid.SIMPLE or DeformableToRigid.INERTIA.
DeformableToRigid.PSET	Identifies the PID type as Part Set. Used for field ptype. Used only for DeformableToRigid.SIMPLE or DeformableToRigid.INERTIA.

Constants for automatic types

Name	Description
DeformableToRigid.D2R	Identifies that card is being written/retreived/removed as D2R card. Used in methods GetDefToRegAutoCard, SetDefToRegAutoCard and RemoveDefToRegAutoCard. Used only for DeformableToRigid.AUTOMATIC.
DeformableToRigid.R2D	Identifies that card is being written/retreived/removed as R2D card. Used in methods GetDefToRegAutoCard, SetDefToRegAutoCard and RemoveDefToRegAutoCard. Used only for DeformableToRigid.AUTOMATIC.

DeformableToRigid properties

Name	Type	Description
code	integer	Activation switch code. (Valid values: 0-5). Used only for DeformableToRigid.AUTOMATIC.
d2r	integer	Number of deformable parts to be switched to rigid plus number of rigid parts for which new lead/constrained rigid body combinations will be defined. Used only for DeformableToRigid.AUTOMATIC.
dtmax	real	Maximum permitted time step size after switch. Used only for DeformableToRigid.AUTOMATIC.
entno	integer	Rigid wall/contact surface number for switch codes 1, 2, 3, 4. Used only for DeformableToRigid.AUTOMATIC.
exists	logical	true if deformable to rigid exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the deformable to rigid is in.
ixx	real	The xx component of inertia tensor. Used only for DeformableToRigid.INERTIA.
ixx	real	The xx component of inertia tensor. Used only for DeformableToRigid.INERTIA.
ixy	real	The xy component of inertia tensor. Used only for DeformableToRigid.INERTIA.
ixz	real	The xz component of inertia tensor. Used only for DeformableToRigid.INERTIA.

iyz	real	The yz component of inertia tensor. Used only for DeformableToRigid.INERTIA.
izz	real	The zz component of inertia tensor. Used only for DeformableToRigid.INERTIA.
lrb	integer	Part ID of the lead rigid body to which the part is merged. Used only for DeformableToRigid.SIMPLE.
model	integer	The Model number that the deformable to rigid is in.
ncsf	integer	Nodal constraint body flag. (Valid values : 0, 1, 2). Used only for DeformableToRigid.AUTOMATIC.
nrbf	integer	Nodal rigid body flag. (Valid values : 0, 1, 2). Used only for DeformableToRigid.AUTOMATIC.
offset	real	Optional contact thickness for switch to deformable. Used only for DeformableToRigid.AUTOMATIC.
paired	integer	Define a pair of related switches. (Valid values : -1, 0, 1). Used only for DeformableToRigid.AUTOMATIC.
pid	integer	Part or Part set ID which is switched to a rigid material. Depends on value of ptype. Used only for DeformableToRigid.SIMPLE or DeformableToRigid.INERTIA.
ptype	integer	Type of PID. Valid values are: DeformableToRigid.PART or DeformableToRigid.PSET. Used only for DeformableToRigid.SIMPLE.
r2d	integer	Number of rigid parts to be switched to deformable. Used only for DeformableToRigid.AUTOMATIC.
relsw	integer	Related switch set. Used only for DeformableToRigid.AUTOMATIC.
rwf	integer	Flag to delete or activate rigid walls. (Valid values : 0, 1, 2). Used only for DeformableToRigid.AUTOMATIC.
swset	integer	Set number for this automatic switch set. (read only). Used only for DeformableToRigid.AUTOMATIC.
time1	real	Switch will not take place before this time. Used only for DeformableToRigid.AUTOMATIC.
time2	real	Switch will not take place after this time. Used only for DeformableToRigid.AUTOMATIC.
time3	real	After this part switch has taken place, another automatic switch will not take place for the duration of the delay period. Used only for DeformableToRigid.AUTOMATIC.
tm	real	Translational mass. Used only for DeformableToRigid.INERTIA.
type	integer	Gives the type of DeformableToRigid Object. (read only)
xc	real	x-coordinate of center of mass. Used only for DeformableToRigid.INERTIA.
yc	real	y-coordinate of center of mass. Used only for DeformableToRigid.INERTIA.
zc	real	z-coordinate of center of mass. Used only for DeformableToRigid.INERTIA.

Detailed Description

The DeformableToRigid class allows you to create, modify, edit and manipulate deformable to rigid cards. See the documentation below for more details.

Constructor

new DeformableToRigid(Model*[Model]*, Type*[constant]*, pid (optional) *[integer]*, lrb (optional)*[integer]*, ptype (optional)*[integer]*)

Description

Create a new DeformableToRigid object.

Arguments

Name	Type	Description
Model	Model	Model that deformable to rigid will be created in
Type	constant	Specify the type of DeformableToRigid (Can be DeformableToRigid.SIMPLE or DeformableToRigid.AUTOMATIC or DeformableToRigid.INERTIA)
pid (optional)	integer	Part or Part set ID which is switched to a rigid material. Depends on value of ptype. Used only for DeformableToRigid.SIMPLE or DeformableToRigid.INERTIA.
lrb (optional)	integer	Part ID of the lead rigid body to which the part is merged. Used only for DeformableToRigid.SIMPLE.
ptype (optional)	integer	Type of PID. Valid values are: DeformableToRigid.PART or DeformableToRigid.PSET. Used only for DeformableToRigid.SIMPLE.

Return type

DeformableToRigid object

Example

To create a new deformable to rigid in model m, type SIMPLE, part id 100:

```
var dtor = new DeformableToRigid(m, DeformableToRigid.SIMPLE, 100);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a deformable to rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the deformable to rigid

Return type

No return value

Example

To associate comment c to the deformable to rigid dtor:

```
dtor.AssociateComment(c);
```

Blank()

Description

Blanks the deformable to rigid

Arguments

No arguments

Return type

No return value

Example

To blank deformable to rigid dtor:

```
dtor.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the deformable to rigids in model m:

```
DeformableToRigid.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged deformable to rigids will be blanked in
flag	Flag	Flag set on the deformable to rigids that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the deformable to rigids in model m flagged with f:

```
DeformableToRigid.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the deformable to rigid is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if deformable to rigid dtor is blanked:

```
if (dtor.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse deformable to rigid dtor:

```
dtor.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the deformable to rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the deformable to rigid

Return type

No return value

Example

To clear flag f for deformable to rigid dtor:

```
dtor.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the deformable to rigid.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

DeformableToRigid object

Example

To copy deformable to rigid dtor into deformable to rigid z:

```
var z = dtor.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an DeformableToRigid definition.

Arguments

Name	Type	Description
Model	Model	Model that the DeformableToRigid will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

DeformableToRigid object (or null if not made)

Example

To start creating an dtor in model m:

```
var dtor = DeformableToRigid.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a deformable to rigid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the deformable to rigid

Return type

No return value

Example

To detach comment c from the deformable to rigid dtor:

```
dtor.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit deformable to rigid dtor:

```
dtor.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for deformable to rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for deformable to rigid dtor:

```
dtor.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first deformable to rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first deformable to rigid in

Return type

DeformableToRigid object (or null if there are no deformable to rigids in the model).

Example

To get the first deformable to rigid in model m:

```
var dtor = DeformableToRigid.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free deformable to rigid label in the model. Also see DeformableToRigid.LastFreeLabel(), DeformableToRigid.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free deformable to rigid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

DeformableToRigid label.

Example

To get the first free deformable to rigid label in model m:

```
var label = DeformableToRigid.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the deformable to rigids in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be flagged in
flag	Flag	Flag to set on the deformable to rigids

Return type

No return value

Example

To flag all of the deformable to rigids with flag f in model m:

```
DeformableToRigid.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the deformable to rigid is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the deformable to rigid

Return type

true if flagged, false if not.

Example

To check if deformable to rigid dtor has flag f set on it:

```
if (dtor.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each deformable to rigid in the model.
Note that ForEach has been designed to make looping over deformable to rigids as fast as possible and so has some limitations.
Firstly, a single temporary DeformableToRigid object is created and on each function call it is updated with the current deformable to rigid data. This means that you should not try to store the DeformableToRigid object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new deformable to rigids inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids are in
func	function	Function to call for each deformable to rigid
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the deformable to rigids in model m:

```
DeformableToRigid.ForEach(m, test);
function test(dtor)
{
// dtor is DeformableToRigid object
}
```

To call function test for all of the deformable to rigids in model m with optional object:

```
var data = { x:0, y:0 };
DeformableToRigid.ForEach(m, test, data);
function test(dtor, extra)
{
// dtor is DeformableToRigid object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of DeformableToRigid objects for all of the deformable to rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get deformable to rigids from

Return type

Array of DeformableToRigid objects

Example

To make an array of DeformableToRigid objects for all of the deformable to rigids in model m

```
var dtor = DeformableToRigid.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a deformable to rigid.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the deformable to rigid dtor:

```
var comm_array = dtor.GetComments();
```

GetDefToRegAutoCard(ctype*[integer]*, index*[integer]*)

Description

Returns the D2R or R2D cards for *DEFORMABLE_TO_RIGID_AUTOMATC.

Arguments

Name	Type	Description
ctype	integer	The card type you want the data for. Can be D2R or R2D.
index	integer	The card index you want the data for. **Note that card indices start at 0, not 1**.

Return type

An array of numbers containing the 2 or 3 member (depending on Card type): Part or Part Set ID, LRB Part ID (only for card type D2R), and part type (PTYPE - Can be DeformableToRigid.PART or DeformableToRigid.PSET).

Example

To get the D2R card data for the 3rd D2R card for Deformable to Rigid dtor:

```
if (dtor.d2r >= 3)
{
    var dtor_data = dtor.GetDefToRegAutoCard(DeformableToRigid.D2R, 2);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of DeformableToRigid objects for all of the flagged deformable to rigids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get deformable to rigids from
flag	Flag	Flag set on the deformable to rigids that you want to retrieve

Return type

Array of DeformableToRigid objects

Example

To make an array of DeformableToRigid objects for all of the deformable to rigids in model m flagged with f

```
var dtor = DeformableToRigid.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the DeformableToRigid object for a deformable to rigid ID.

Arguments

Name	Type	Description
Model	Model	Model to find the deformable to rigid in
number	integer	number of the deformable to rigid you want the DeformableToRigid object for

Return type

DeformableToRigid object (or null if deformable to rigid does not exist).

Example

To get the DeformableToRigid object for deformable to rigid 100 in model m

```
var dtor = DeformableToRigid.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a DeformableToRigid property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the DeformableToRigid.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	deformable to rigid property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if DeformableToRigid property dtor.example is a parameter:

```
Options.property_parameter_names = true;
if (dtor.GetParameter(dtor.example) ) do_something...
Options.property_parameter_names = false;
```

To check if DeformableToRigid property dtor.example is a parameter by using the GetParameter method:

```
if (dtor.ViewParameters().GetParameter(dtor.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this DeformableToRigid (*DEFORMABLE_TO_RIGID_xxxx) **Note that a carriage return is not added**. See also DeformableToRigid.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for DeformableToRigid dtor:

```
var key = dtor.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the DeformableToRigid. **Note that a carriage return is not added**. See also DeformableToRigid.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for DeformableToRigid dtor:

```
var cards = dtor.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last deformable to rigid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last deformable to rigid in

Return type

DeformableToRigid object (or null if there are no deformable to rigids in the model).

Example

To get the last deformable to rigid in model m:

```
var dtor = DeformableToRigid.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free deformable to rigid label in the model. Also see DeformableToRigid.FirstFreeLabel(), DeformableToRigid.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free deformable to rigid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

DeformableToRigid label.

Example

To get the last free deformable to rigid label in model m:

```
var label = DeformableToRigid.LastFreeLabel(m);
```

Next()

Description

Returns the next deformable to rigid in the model.

Arguments

No arguments

Return type

DeformableToRigid object (or null if there are no more deformable to rigids in the model).

Example

To get the deformable to rigid in model m after deformable to rigid dtor:

```
var dtor = dtor.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) deformable to rigid label in the model. Also see DeformableToRigid.FirstFreeLabel(), DeformableToRigid.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free deformable to rigid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

DeformableToRigid label.

Example

To get the next free deformable to rigid label in model m:

```
var label = DeformableToRigid.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a deformable to rigid.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only deformable to rigids from that model can be picked. If the argument is a Flag then only deformable to rigids that are flagged with *limit* can be selected. If omitted, or null, any deformable to rigids from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

DeformableToRigid object (or null if not picked)

Example

To pick a deformable to rigid from model m giving the prompt 'Pick deformable to rigid from screen':

```
var dtor = DeformableToRigid.Pick('Pick deformable to rigid from screen', m);
```

Previous()

Description

Returns the previous deformable to rigid in the model.

Arguments

No arguments

Return type

DeformableToRigid object (or null if there are no more deformable to rigids in the model).

Example

To get the deformable to rigid in model m before deformable to rigid dtor:

```
var dtor = dtor.Previous();
```

RemoveDefToRegAutoCard(ctype*[integer]*, index*[integer]*)

Description

Removes the D2R or R2D cards for *DEFORMABLE_TO_RIGID_AUTOMATC.

Arguments

Name	Type	Description
ctype	integer	The card type you want removed. Can be D2R or R2D.
index	integer	The card index you want removed. **Note that card indices start at 0, not 1**.

Return type

No return value.

Example

To remove the D2R card data for the 3rd D2R card from Deformable to Rigid dtor:

```
if (dtor.d2r >= 3)
{
    var dtor_data = dtor.RemoveDefToRegAutoCard(DeformableToRigid.D2R, 2);
}
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the deformable to rigids in model m, from 1000000:

```
DeformableToRigid.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged deformable to rigids will be renumbered in
flag	Flag	Flag set on the deformable to rigids that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the deformable to rigids in model m flagged with f, from 1000000:

```
DeformableToRigid.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select deformable to rigids using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting deformable to rigids
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only deformable to rigids from that model can be selected. If the argument is a Flag then only deformable to rigids that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any deformable to rigids can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of deformable to rigids selected or null if menu cancelled

Example

To select deformable to rigids from model m, flagging those selected with flag f, giving the prompt 'Select deformable to rigids':

```
DeformableToRigid.Select(f, 'Select deformable to rigids', m);
```

To select deformable to rigids, flagging those selected with flag f but limiting selection to deformable to rigids flagged with flag l, giving the prompt 'Select deformable to rigids':

```
DeformableToRigid.Select(f, 'Select deformable to rigids', l);
```

SetDefToRegAutoCard(ctype*[integer]*, index*[integer]*, ptype*[integer]*, pid*[integer]*, lrb (optional)*[integer]*)

Description

Sets the D2r or R2D card data f*DEFORMABLE_TO_RIGID_AUTOMATIC.

Arguments

Name	Type	Description
ctype	integer	The card type you want to set. Can be D2R or R2D.
index	integer	The D2R or R2D card index you want to set. **Note that cards start at 0, not 1**.
ptype	integer	Part type (PTYPE). Can be DeformableToRigid.PART or DeformableToRigid.PSET.
pid	integer	Part or Part Set ID.
lrb (optional)	integer	LRB Part ID (only for card type D2R)

Return type

No return value.

Example

To set the 3rd D2R card to ptype DeformabletoRigid.PART, pid 100 and lrb 200, for DeformableToRigid dtor:

```
dtor.SetDefToRegAutoCard(DeformabletoRigid.D2R, 2, DeformabletoRigid.PART, 100,
200);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the deformable to rigid.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the deformable to rigid

Return type

No return value

Example

To set flag f for deformable to rigid dtor:

```
dtor.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the deformable to rigid. The deformable to rigid will be sketched until you either call DeformableToRigid.Unsketch(), DeformableToRigid.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the deformable to rigid is sketched. If omitted redraw is true. If you want to sketch several deformable to rigids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch deformable to rigid dtor:

```
dtor.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged deformable to rigids in the model. The deformable to rigids will be sketched until you either call DeformableToRigid.Unsketch(), DeformableToRigid.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged deformable to rigids will be sketched in
flag	Flag	Flag set on the deformable to rigids that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the deformable to rigids are sketched. If omitted redraw is true. If you want to sketch flagged deformable to rigids several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all deformable to rigids flagged with flag in model m:

```
DeformableToRigid.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of deformable to rigids in the model.

Arguments

Name	Type	Description

Model	Model	Model to get total for
exists (optional)	boolean	true if only existing deformable to rigids should be counted. If false or omitted referenced but undefined deformable to rigids will also be included in the total.

Return type

number of deformable to rigids

Example

To get the total number of deformable to rigids in model m:

```
var total = DeformableToRigid.Total(m);
```

Unblank()

Description

Unblanks the deformable to rigid

Arguments

No arguments

Return type

No return value

Example

To unblank deformable to rigid dtor:

```
dtor.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the deformable to rigids in model m:

```
DeformableToRigid.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged deformable to rigids will be unblanked in
flag	Flag	Flag set on the deformable to rigids that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the deformable to rigids in model m flagged with f:

```
DeformableToRigid.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all deformable to rigids will be unset in
flag	Flag	Flag to unset on the deformable to rigids

Return type

No return value

Example

To unset the flag f on all the deformable to rigids in model m:

```
DeformableToRigid.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the deformable to rigid.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the deformable to rigid is unsketched. If omitted redraw is true. If you want to unsketch several deformable to rigids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch deformable to rigid dtor:

```
dtor.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all deformable to rigids.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the deformable to rigids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all deformable to rigids in model m:

```
DeformableToRigid.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged deformable to rigids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all deformable to rigids will be unsketched in
flag	Flag	Flag set on the deformable to rigids that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the deformable to rigids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all deformable to rigids flagged with flag in model m:

```
DeformableToRigid.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

DeformableToRigid object.

Example

To check if DeformableToRigid property dtor.example is a parameter by using the DeformableToRigid.GetParameter() method:

```
if (dtor.ViewParameters().GetParameter(dtor.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for deformable to rigid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for deformable to rigid dtor:

```
dtor.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this deformable to rigid.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for deformable to rigid dtor:

```
var xrefs = dtor.Xrefs();
```

---

toString()

Description

Creates a string containing the DeformableToRigid data in keyword format. Note that this contains the keyword header and the keyword cards. See also DeformableToRigid.Keyword() and DeformableToRigid.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for DeformableToRigid dtor in keyword format

```
var i_str = dtor.toString();
```

Accelerometer class

The Accelerometer class gives you access to seatbelt accelerometer cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Accelerometer properties

Name	Type	Description
colour	Colour	The colour of the accelerometer
exists	logical	true if accelerometer exists, false if referred to but not defined. (read only)
igrav	integer	Gravitational acceleration due to body force loads is included in acceleration output if igrav is 0, removed if igrav is 1.
include	integer	The Include file number that the accelerometer is in.
intopt	integer	Integration option; velocities are integrated from global accelerations and transformed into local system if intopt is 0, they are integrated directly from local accelerations if intopt is 1.
label	integer	Accelerometer number. Also see the sbacid property which is an alternative name for this.
mass	real	Optional added mass for accelerometer
model	integer	The Model number that the accelerometer is in.
nid1	integer	Node number 1
nid2	integer	Node number 2
nid3	integer	Node number 3
sbacid	integer	Accelerometer number. Also see the label property which is an alternative name for this.
transparency	integer	The transparency of the accelerometer (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Accelerometer class allows you to create, modify, edit and manipulate seatbelt accelerometer cards. See the documentation below for more details.

Constructor

new Accelerometer(Model*[Model]*, sbacid*[integer]*, nid1*[integer]*, nid2*[integer]*, nid3*[integer]*, igrav (optional)*[integer]*, intopt (optional)*[integer]*, mass (optional)*[real]*)

Description

Create a new Seatbelt Accelerometer object.

Arguments

Name	Type	Description
Model	Model	Model that accelerometer will be created in
sbacid	integer	Accelerometer number. Also see the label property which is an alternative name for this.
nid1	integer	Node number 1
nid2	integer	Node number 2
nid3	integer	Node number 3
igrav (optional)	integer	Gravitational acceleration due to body force loads is included in acceleration output if igrav is 0, removed if igrav is 1.
intopt (optional)	integer	Integration option; velocities are integrated from global accelerations and transformed into local system if intopt is 0, they are integrated directly from local accelerations if intopt is 1.

mass (optional)	real	Optional added mass for accelerometer

Return type

Accelerometer object

Example

To create a new seatbelt accelerometer in model m with label 100, nodes 1, 2 and 3:

```
var a = new Accelerometer(m, 100, 1, 2, 3);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a accelerometer.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the accelerometer

Return type

No return value

Example

To associate comment c to the accelerometer a:

```
a.AssociateComment(c);
```

Blank()

Description

Blanks the accelerometer

Arguments

No arguments

Return type

No return value

Example

To blank accelerometer a:

```
a.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the accelerometers in model m:

```
Accelerometer.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged accelerometers will be blanked in
flag	Flag	Flag set on the accelerometers that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the accelerometers in model m flagged with f:

```
Accelerometer.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the accelerometer is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if accelerometer a is blanked:

```
if (a.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse accelerometer a:

```
a.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the accelerometer.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the accelerometer

Return type

No return value

Example

To clear flag f for accelerometer a:

```
a.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the accelerometer.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Accelerometer object

Example

To copy accelerometer a into accelerometer z:

```
var z = a.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a accelerometer.

Arguments

Name	Type	Description
Model	Model	Model that the accelerometer will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Accelerometer object (or null if not made)

Example

To start creating an accelerometer in model m:

```
var a = Accelerometer.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a accelerometer.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the accelerometer

Return type

No return value

Example

To detach comment c from the accelerometer a:

```
a.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit accelerometer a:

```
a.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for accelerometer. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for accelerometer a:

```
a.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for accelerometer.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the accelerometer colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the accelerometer.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing accelerometer a:

```
var colour = a.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first accelerometer in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first accelerometer in

Return type

Accelerometer object (or null if there are no accelerometers in the model).

Example

To get the first accelerometer in model m:

```
var a = Accelerometer.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free accelerometer label in the model. Also see Accelerometer.LastFreeLabel(), Accelerometer.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free accelerometer label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Accelerometer label.

Example

To get the first free accelerometer label in model m:

```
var label = Accelerometer.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the accelerometers in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be flagged in
flag	Flag	Flag to set on the accelerometers

Return type

No return value

Example

To flag all of the accelerometers with flag f in model m:

```
Accelerometer.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the accelerometer is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the accelerometer

Return type

true if flagged, false if not.

Example

To check if accelerometer a has flag f set on it:

```
if (a.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each accelerometer in the model.
Note that ForEach has been designed to make looping over accelerometers as fast as possible and so has some limitations.
Firstly, a single temporary Accelerometer object is created and on each function call it is updated with the current accelerometer data. This means that you should not try to store the Accelerometer object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new accelerometers inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers are in
func	function	Function to call for each accelerometer
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the accelerometers in model m:

```
Accelerometer.ForEach(m, test);
function test(a)
{
// a is Accelerometer object
}
```

To call function test for all of the accelerometers in model m with optional object:

```
var data = { x:0, y:0 };
Accelerometer.ForEach(m, test, data);
function test(a, extra)
{
// a is Accelerometer object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Accelerometer objects for all of the accelerometers in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get accelerometers from

Return type

Array of Accelerometer objects

Example

To make an array of Accelerometer objects for all of the accelerometers in model m

```
var a = Accelerometer.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a accelerometer.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the accelerometer a:

```
var comm_array = a.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Accelerometer objects for all of the flagged accelerometers in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get accelerometers from
flag	Flag	Flag set on the accelerometers that you want to retrieve

Return type

Array of Accelerometer objects

Example

To make an array of Accelerometer objects for all of the accelerometers in model m flagged with f

```
var a = Accelerometer.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Accelerometer object for a accelerometer ID.

Arguments

Name	Type	Description
Model	Model	Model to find the accelerometer in
number	integer	number of the accelerometer you want the Accelerometer object for

Return type

Accelerometer object (or null if accelerometer does not exist).

Example

To get the Accelerometer object for accelerometer 100 in model m

```
var a = Accelerometer.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Accelerometer property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Accelerometer.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	accelerometer property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Accelerometer property a.example is a parameter:

```
Options.property_parameter_names = true;
if (a.GetParameter(a.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Accelerometer property a.example is a parameter by using the GetParameter method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this accelerometer (*ELEMENT_SEATBELT_ACCELEROMETER) **Note that a carriage return is not added**. See also Accelerometer.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for accelerometer a:

```
var key = a.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the accelerometer. **Note that a carriage return is not added**. See also Accelerometer.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for accelerometer a:

```
var cards = a.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last accelerometer in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last accelerometer in

Return type

Accelerometer object (or null if there are no accelerometers in the model).

Example

To get the last accelerometer in model m:

```
var a = Accelerometer.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free accelerometer label in the model. Also see Accelerometer.FirstFreeLabel(), Accelerometer.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free accelerometer label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Accelerometer label.

Example

To get the last free accelerometer label in model m:

```
var label = Accelerometer.LastFreeLabel(m);
```

Next()

Description

Returns the next accelerometer in the model.

Arguments

No arguments

Return type

Accelerometer object (or null if there are no more accelerometers in the model).

Example

To get the accelerometer in model m after accelerometer a:

```
var a = a.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) accelerometer label in the model. Also see Accelerometer.FirstFreeLabel(), Accelerometer.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free accelerometer label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Accelerometer label.

Example

To get the next free accelerometer label in model m:

```
var label = Accelerometer.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a accelerometer.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only accelerometers from that model can be picked. If the argument is a Flag then only accelerometers that are flagged with *limit* can be selected. If omitted, or null, any accelerometers from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Accelerometer object (or null if not picked)

Example

To pick a accelerometer from model m giving the prompt 'Pick accelerometer from screen':

```
var a = Accelerometer.Pick('Pick accelerometer from screen', m);
```

Previous()

Description

Returns the previous accelerometer in the model.

Arguments

No arguments

Return type

Accelerometer object (or null if there are no more accelerometers in the model).

Example

To get the accelerometer in model m before accelerometer a:

```
var a = a.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the accelerometers in model m, from 1000000:

```
Accelerometer.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged accelerometers will be renumbered in
flag	Flag	Flag set on the accelerometers that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the accelerometers in model m flagged with f, from 1000000:

```
Accelerometer.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select accelerometers using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting accelerometers
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only accelerometers from that model can be selected. If the argument is a Flag then only accelerometers that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any accelerometers can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of accelerometers selected or null if menu cancelled

Example

To select accelerometers from model m, flagging those selected with flag f, giving the prompt 'Select accelerometers':

```
Accelerometer.Select(f, 'Select accelerometers', m);
```

To select accelerometers, flagging those selected with flag f but limiting selection to accelerometers flagged with flag l, giving the prompt 'Select accelerometers':

```
Accelerometer.Select(f, 'Select accelerometers', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the accelerometer.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the accelerometer

Return type

No return value

Example

To set flag f for accelerometer a:

```
a.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the accelerometer. The accelerometer will be sketched until you either call Accelerometer.Unsketch(), Accelerometer.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the accelerometer is sketched. If omitted redraw is true. If you want to sketch several accelerometers and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch accelerometer a:

```
a.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged accelerometers in the model. The accelerometers will be sketched until you either call Accelerometer.Unsketch(), Accelerometer.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged accelerometers will be sketched in
flag	Flag	Flag set on the accelerometers that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the accelerometers are sketched. If omitted redraw is true. If you want to sketch flagged accelerometers several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all accelerometers flagged with flag in model m:

```
Accelerometer.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing accelerometers should be counted. If false or omitted referenced but undefined accelerometers will also be included in the total.

Return type

number of accelerometers

Example

To get the total number of accelerometers in model m:

```
var total = Accelerometer.Total(m);
```

Unblank()

Description

Unblanks the accelerometer

Arguments

No arguments

Return type

No return value

Example

To unblank accelerometer a:

```
a.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the accelerometers in model m:

```
Accelerometer.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged accelerometers will be unblanked in
flag	Flag	Flag set on the accelerometers that you want to unblank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the accelerometers in model m flagged with f:

```
Accelerometer.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all accelerometers will be unset in
flag	Flag	Flag to unset on the accelerometers

Return type

No return value

Example

To unset the flag f on all the accelerometers in model m:

```
Accelerometer.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the accelerometer.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the accelerometer is unsketched. If omitted redraw is true. If you want to unsketch several accelerometers and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch accelerometer a:

```
a.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all accelerometers.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the accelerometers are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all accelerometers in model m:

```
Accelerometer.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged accelerometers in the model.

Arguments

Name	Type	Description
Model	Model	Model that all accelerometers will be unsketched in
flag	Flag	Flag set on the accelerometers that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the accelerometers are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all accelerometers flagged with flag in model m:

```
Accelerometer.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Accelerometer object.

Example

To check if Accelerometer property a.example is a parameter by using the Accelerometer.GetParameter() method:

```
if (a.ViewParameters().GetParameter(a.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for accelerometer. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for accelerometer a:

```
a.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this accelerometer.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for accelerometer a:

```
var xrefs = a.Xrefs();
```

---

toString()

Description

Creates a string containing the accelerometer data in keyword format. Note that this contains the keyword header and the keyword cards. See also Accelerometer.Keyword() and Accelerometer.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for accelerometer a in keyword format

```
var str = a.toString();
```

Beam class

The Beam class gives you access to beam cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- FindBeamEnd() [deprecated]
- FindBeamInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, bflag (optional)*[integer]*, bthick (optional)*[integer]*)
- FindBeamInit(Model*[Model]*, flag (optional)*[Flag]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- ElemCut(Database cross section label*[integer]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SectionFacePoints(face*[integer]*)
- SectionFaces()

- SectionPoints()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Beam properties

Name	Type	Description
cid	integer	Coordinate system ID (_SCALAR)
cid	integer	Coordinate system ID at node 1 (_SCALR)
colour	Colour	The colour of the beam
d1	real	Section parameter 1
d2	real	Section parameter 2
d3	real	Section parameter 3
d4	real	Section parameter 4
d5	real	Section parameter 5
d6	real	Section parameter 6
dofn1	integer	Active degree of freedom at node 1 (_SCALAR)
dofn2	integer	Active degree of freedom at node 2 (_SCALAR)
dofns	integer	Active degrees of freedom at nodes 1 and 2 (_SCALR)
eid	integer	Beam number. Also see the label property which is an alternative name for this.
elbow	logical	If ELBOW option is set. Can be true or false
exists	logical	true if beam exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the beam is in.
iner	real	Mass moment of inertia for beam
label	integer	Beam number. Also see the eid property which is an alternative name for this.
local	integer	Coordinate system option
mn	integer	Middle Node for elbow beam
model	integer	The Model number that the beam is in.
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
nodes	integer	Number of nodes beam has (read only)
offset	real	If _OFFSET option is set. Can be true or false
orientation	real	If _ORIENTATION option is set. Can be true or false
parm1	real	Thickness parameter 1
parm2	real	Thickness parameter 2

parm3	real	Thickness parameter 3
parm4	real	Thickness parameter 4
parm5	real	Thickness parameter 5
pid	integer	Part number
pid1	integer	Part number 1 for spotweld beam
pid2	integer	Part number 2 for spotweld beam
pid_opt	logical	If _PID option is set. Can be true or false
rr1	integer	Rotational release code at node 1
rr2	integer	Rotational release code at node 2
rt1	integer	Translational release code at node 1
rt2	integer	Translational release code at node 2
scalar	logical	If _SCALAR option is set. Can be true or false
scalr	logical	If _SCALR option is set. Can be true or false
section	logical	If _SECTION option is set. Can be true or false
sn1	integer	Scalar Node number 1
sn2	integer	Scalar Node number 2
stype	string	Section type
thickness	logical	If _THICKNESS option is set. Can be true or false
transparency	integer	The transparency of the beam (0-100) 0% is opaque, 100% is transparent.
vol	real	Volume of beam
vx	real	Orientation vector X at node 1
vy	real	Orientation vector Y at node 1
vz	real	Orientation vector Z at node 1
warpage	logical	If WARPAGE option is set. Can be true or false
wx1	real	Offset vector X at node 1
wx2	real	Offset vector X at node 2
wy1	real	Offset vector Y at node 1
wy2	real	Offset vector Y at node 2
wz1	real	Offset vector Z at node 1
wz2	real	Offset vector Z at node 2

Detailed Description

The Beam class allows you to create, modify, edit and manipulate beam cards. See the documentation below for more details.

Constructor

new Beam(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*)

Description

Create a new Beam object. Use either 1, 2 or 3 nodes when creating a new beam.

Arguments

Name	Type	Description
Model	Model	Model that beam will be created in
eid	integer	Beam number
pid	integer	Part number
n1	integer	Node number 1
n2	integer	Node number 2 (optional)
n3	integer	Node number 3 (optional)

Return type

Beam object

Example

To create a new beam in model m with label 100, part 10 and nodes 1, 2, 3:

```
var b = new Beam(m, 100, 10, 1, 2, 3);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the beam

Return type

No return value

Example

To associate comment c to the beam b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the beam

Arguments

No arguments

Return type

No return value

Example

To blank beam b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the beams in model m:

```
Beam.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged beams will be blanked in
flag	Flag	Flag set on the beams that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the beams in model m flagged with f:

```
Beam.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the beam is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if beam b is blanked:

```
if (b.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse beam b:

```
b.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the beam.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the beam

Return type

No return value

Example

To clear flag f for beam b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the beam.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Beam object

Example

To copy beam b into beam z:

```
var z = b.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a beam.

Arguments

Name	Type	Description
Model	Model	Model that the beam will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Beam object (or null if not made)

Example

To start creating a beam in model m:

```
var s = Beam.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the beam

Return type

No return value

Example

To detach comment c from the beam b:

```
b.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit beam b:

```
b.Edit();
```

ElemCut(Database cross section label*[integer]*)

Description

Returns coordinates of the intersections between a beam and a database cross section.

Arguments

Name	Type	Description
Database cross section label	integer	The label of the database cross section.

Return type

An array containing the x,y,z coordinates of the cut point, or NULL if it does not cut. Note this function does not check that the beam is in the cross section definition (part set)

Example

To get the cut line coordinates between database cross section 200 and beam b:

```
var data = b.ElemCut(200)
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for beam b:

```
b.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for beam.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the beam colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the beam.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing beam b:

```
var colour = b.ExtractColour();
```

FindBeamEnd() [static] [deprecated]

This function is deprecated in version 16.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Tidy memory allocation incurred by function which finds beams within a box. Now replaced by model member function Model.FindElemEnd()

Arguments

No arguments

Return type

No return value

Example

```
Beam.FindBeamEnd();
```

FindBeamInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, bflag (optional)*[integer]*, bthick (optional)*[integer]*) [static]

Description

Returns an array of Beam objects for the beams within a box. This requires a previous (outside loop) call to function FindBeamInit() or m.FindElemInit() where the process is initialized for flagged beams in the model (typically all beams) and m.FindElemEnd() to close the process. Please note this function provides a list of all beams that could potentially be in the box (using computationally cheap bounding box comparison) it is not a rigorous test of whether the beam is actually in the box. See also Beam.FindBeamInit() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model designated model
xmin	real	Minimum bound in global x
xmax	real	Maximum bound in global x
ymin	real	Minimum bound in global y
ymax	real	Maximum bound in global y
zmin	real	Minimum bound in global z
zmax	real	Maximum bound in global z
bflag (optional)	integer	Optional flag to restrict beams considered
bthick (optional)	integer	Optional flag to consider thickness for beams

Return type

Array of Beam objects

Example

To get an array of Beam objects for beams in model m within defined box

```
Beam.FindBeamInit(m);
```

or clear model flag and flag elements of interest and

```
m.FindElemInit(flag);

{

    //loop in which boxes are formed and tested

    //find beams both in box and flagged with bflag

    //consider beam thickness

    var s = Beam.FindBeamInBox(m, xmin, xmax, ymin, ymax, zmin, zmax, bflag, 1);

    if(s.length) ...

}

m.FindElemEnd();
```

FindBeamInit(Model*[Model]*, flag (optional)*[Flag]*) [static]

Description

Initialize setup so that all flagged beams in model can be tested to see if they are within box. See also Beam.FindBeamInBox() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model in which beams have been flagged
flag (optional)	Flag	Optional flag that has been set on the beams, if unsupplied all beams considered

Return type

No return value

Example

To initialize find setup for all beams in model m:

```
Beam.FindBeamInit(m);
```

First(Model*[Model]*) [static]

Description

Returns the first beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first beam in

Return type

Beam object (or null if there are no beams in the model).

Example

To get the first beam in model m:

```
var b = Beam.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free beam label in the model. Also see Beam.LastFreeLabel(), Beam.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Beam label.

Example

To get the first free beam label in model m:

```
var label = Beam.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the beams in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be flagged in
flag	Flag	Flag to set on the beams

Return type

No return value

Example

To flag all of the beams with flag f in model m:

```
Beam.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the beam is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the beam

Return type

true if flagged, false if not.

Example

To check if beam b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each beam in the model.
Note that ForEach has been designed to make looping over beams as fast as possible and so has some limitations. Firstly, a single temporary Beam object is created and on each function call it is updated with the current beam data. This means that you should not try to store the Beam object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new beams inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all beams are in
func	function	Function to call for each beam
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the beams in model m:

```
Beam.ForEach(m, test);
function test(b)
{
// b is Beam object
}
```

To call function test for all of the beams in model m with optional object:

```
var data = { x:0, y:0 };
Beam.ForEach(m, test, data);
function test(b, extra)
{
// b is Beam object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Beam objects for all of the beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get beams from

Return type

Array of Beam objects

Example

To make an array of Beam objects for all of the beams in model m

```
var b = Beam.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a beam.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the beam b:

```
var comm_array = b.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Beam objects for all of the flagged beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get beams from
flag	Flag	Flag set on the beams that you want to retrieve

Return type

Array of Beam objects

Example

To make an array of Beam objects for all of the beams in model m flagged with f

```
var b = Beam.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Beam object for a beam ID.

Arguments

Name	Type	Description
Model	Model	Model to find the beam in
number	integer	number of the beam you want the Beam object for

Return type

Beam object (or null if beam does not exist).

Example

To get the Beam object for beam 100 in model m

```
var b = Beam.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Beam property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Beam.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	beam property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Beam property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Beam property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this beam (*BEAM, *BEAM_SCALAR or *BEAM_SCALAR_VALUE). **Note that a carriage return is not added**. See also Beam.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for beam s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the beam. **Note that a carriage return is not added**. See also Beam.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for beam b:

```
var cards = b.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last beam in

Return type

Beam object (or null if there are no beams in the model).

Example

To get the last beam in model m:

```
var b = Beam.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free beam label in the model. Also see Beam.FirstFreeLabel(), Beam.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Beam label.

Example

To get the last free beam label in model m:

```
var label = Beam.LastFreeLabel(m);
```

Next()

Description

Returns the next beam in the model.

Arguments

No arguments

Return type

Beam object (or null if there are no more beams in the model).

Example

To get the beam in model m after beam b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) beam label in the model. Also see Beam.FirstFreeLabel(), Beam.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Beam label.

Example

To get the next free beam label in model m:

```
var label = Beam.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a beam.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only beams from that model can be picked. If the argument is a Flag then only beams that are flagged with *limit* can be selected. If omitted, or null, any beams from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Beam object (or null if not picked)

Example

To pick a beam from model m giving the prompt 'Pick beam from screen':

```
var b = Beam.Pick('Pick beam from screen', m);
```

Previous()

Description

Returns the previous beam in the model.

Arguments

No arguments

Return type

Beam object (or null if there are no more beams in the model).

Example

To get the beam in model m before beam b:

```
var b = b.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the beams in model m, from 1000000:

```
Beam.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged beams will be renumbered in
flag	Flag	Flag set on the beams that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the beams in model m flagged with f, from 1000000:

```
Beam.RenumberFlagged(m, f, 1000000);
```

SectionFacePoints(face*[integer]*)

Description

Returns the indices of the points for a faces to plot the true section of the beam. Note face numbers start at 0. Beam.SectionPoints must be called before this method.

Arguments

Name	Type	Description
face	integer	Face to get indices for

Return type

Array of integers

Example

To get the indices of the points for the second face on beam b:

```
var indices = b.SectionFacePoints(1);
```

SectionFaces()

Description

Returns the number of faces to plot the true section of the beam. Beam.SectionPoints must be called before this method.

Arguments

No arguments

Return type

integer

Example

To get the number of faces for beam b:

```
var faces = b.SectionFaces();
```

SectionPoints()

Description

Returns the point coordinates to plot the true section of the beam. They are returned in a single array of numbers.

Arguments

No arguments

Return type

Array of reals

Example

To get the point coordinates for beam b:

```
var points = b.SectionPoints();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select beams using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting beams
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only beams from that model can be selected. If the argument is a Flag then only beams that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any beams can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of beams selected or null if menu cancelled

Example

To select beams from model m, flagging those selected with flag f, giving the prompt 'Select beams':

```
Beam.Select(f, 'Select beams', m);
```

To select beams, flagging those selected with flag f but limiting selection to beams flagged with flag l, giving the prompt 'Select beams':

```
Beam.Select(f, 'Select beams', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the beam.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the beam

Return type

No return value

Example

To set flag f for beam b:

```
b.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the beam. The beam will be sketched until you either call Beam.Unsketch(), Beam.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the beam is sketched. If omitted redraw is true. If you want to sketch several beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch beam b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged beams in the model. The beams will be sketched until you either call Beam.Unsketch(), Beam.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged beams will be sketched in
flag	Flag	Flag set on the beams that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the beams are sketched. If omitted redraw is true. If you want to sketch flagged beams several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all beams flagged with flag in model m:

```
Beam.SketchFlagged(m, flag);
```

TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)

Description

Checks if nodes of beam are tied by contact or directly attached (non-zero option1)

Arguments

Name	Type	Description
Contact label	integer	The label of the tied contact. If zero the tied contact is found for the beam by reverse lookup.
Flag	Flag	flag bit
Option1	integer	Directly tied node (logical OR) 0:NONE 1:NRB/C_EXNO 2:BEAM 4:SHELL 8:SOLID 16:TSHELL
Option2	integer	0:No action 1:report error if directly attached node (acc. option1) also captured by contact

Return type

string

Example

To check if both nodes of beam b are tied by contact 200 or attach directly to constraint, beam or shell:

```
var message = b.TiedNodeCheck(200, flag, 1|2|4, 1)
```

Timestep()

Description

Calculates the timestep for the beam

Arguments

No arguments

Return type

real

Example

To calculate the timestep for beam b:

```
var timestep = b.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of beams in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing beams should be counted. If false or omitted referenced but undefined beams will also be included in the total.

Return type

number of beams

Example

To get the total number of beams in model m:

```
var total = Beam.Total(m);
```

Unblank()

Description

Unblanks the beam

Arguments

No arguments

Return type

No return value

Example

To unblank beam b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the beams in model m:

```
Beam.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged beams will be unblanked in
flag	Flag	Flag set on the beams that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the beams in model m flagged with f:

```
Beam.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all beams will be unset in
flag	Flag	Flag to unset on the beams

Return type

No return value

Example

To unset the flag f on all the beams in model m:

```
Beam.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the beam.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the beam is unsketched. If omitted redraw is true. If you want to unsketch several beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch beam b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all beams.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all beams in model m:

```
Beam.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all beams will be unsketched in
flag	Flag	Flag set on the beams that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all beams flagged with flag in model m:

```
Beam.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Beam object.

Example

To check if Beam property b.example is a parameter by using the Beam.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for beam b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this beam.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for beam b:

```
var xrefs = b.Xrefs();
```

---

toString()

Description

Creates a string containing the beam data in keyword format. Note that this contains the keyword header and the keyword cards. See also Beam.Keyword() and Beam.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for beam b in keyword format

```
var str = b.toString();
```

Discrete class

The Discrete class gives you access to element discrete cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)

- Xrefs()
- toString()

Discrete properties

Name	Type	Description
colour	Colour	The colour of the discrete
eid	integer	Discrete number. Also see the label property which is an alternative name for this.
exists	logical	true if discrete exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the discrete is in.
label	integer	Discrete number. Also see the eid property which is an alternative name for this.
lcid	integer	Loadcurve for offset vs time
lciddr	integer	Loadcurve for offset vs time during dynamic relaxation
lco	boolean	If LCO option is set. Can be true or false
model	integer	The Model number that the discrete is in.
n1	integer	Node number 1
n2	integer	Node number 2
offset	real	Initial offset
pf	integer	Print flag. Set to write forces to the DEFORC file
pid	integer	Part number
s	real	Scale factor on forces
transparency	integer	The transparency of the discrete (0-100) 0% is opaque, 100% is transparent.
vid	integer	Orientation vector

Detailed Description

The Discrete class allows you to create, modify, edit and manipulate discrete cards. See the documentation below for more details.

Constructor

new Discrete(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, vid (optional)*[integer]*, s (optional)*[real]*, pf (optional)*[integer]*, offset (optional)*[real]*)

Description

Create a new Discrete object.

Arguments

Name	Type	Description
Model	Model	Model that discrete will be created in
eid	integer	Discrete number
pid	integer	Part number
n1	integer	Node number 1
n2	integer	Node number 2

vid (optional)	integer	Orientation vector
s (optional)	real	Scale factor on forces
pf (optional)	integer	Print flag. Set to write forces to the DEFORC file
offset (optional)	real	Initial offset

Return type

Discrete object

Example

To create a new discrete in model m with label 200, in part 10, on nodes 1 and 2

```
var m = new Discrete(m, 200, 10, 1, 2);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a discrete.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the discrete

Return type

No return value

Example

To associate comment c to the discrete d:

```
d.AssociateComment(c);
```

Blank()

Description

Blanks the discrete

Arguments

No arguments

Return type

No return value

Example

To blank discrete d:

```
d.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the discretes in model m:

```
Discrete.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged discretes will be blanked in
flag	Flag	Flag set on the discretes that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the discretes in model m flagged with f:

```
Discrete.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the discrete is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if discrete d is blanked:

```
if (d.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse discrete d:

```
d.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the discrete.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the discrete

Return type

No return value

Example

To clear flag f for discrete d:

```
d.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the discrete.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Discrete object

Example

To copy discrete d into discrete z:

```
var z = d.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a discrete.

Arguments

Name	Type	Description
Model	Model	Model that the discrete will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Discrete object (or null if not made)

Example

To start creating a discrete in model m:

```
var m = Discrete.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a discrete.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the discrete

Return type

No return value

Example

To detach comment c from the discrete d:

```
d.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit discrete d:

```
d.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for discrete. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for discrete d:

```
d.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for discrete.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the discrete colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the discrete.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing discrete d:

```
var colour = d.ExtractColour();
```

First(Model[*Model*]) [static]

Description

Returns the first discrete in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first discrete in

Return type

Discrete object (or null if there are no discretes in the model).

Example

To get the first discrete in model m:

```
var d = Discrete.First(m);
```

FirstFreeLabel(Model[*Model*], layer (optional)[*Include number*]) [static]

Description

Returns the first free discrete label in the model. Also see Discrete.LastFreeLabel(), Discrete.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free discrete label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Discrete label.

Example

To get the first free discrete label in model m:

```
var label = Discrete.FirstFreeLabel(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the discretes in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be flagged in
flag	Flag	Flag to set on the discretes

Return type

No return value

Example

To flag all of the discretes with flag f in model m:

```
Discrete.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the discrete is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the discrete

Return type

true if flagged, false if not.

Example

To check if discrete d has flag f set on it:

```
if (d.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each discrete in the model.
Note that ForEach has been designed to make looping over discretes as fast as possible and so has some limitations.
Firstly, a single temporary Discrete object is created and on each function call it is updated with the current discrete data. This means that you should not try to store the Discrete object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new discretes inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all discretes are in
func	function	Function to call for each discrete
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the discretes in model m:

```
Discrete.ForEach(m, test);
function test(d)
{
// d is Discrete object
}
```

To call function test for all of the discretes in model m with optional object:

```
var data = { x:0, y:0 };
Discrete.ForEach(m, test, data);
function test(d, extra)
{
// d is Discrete object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Discrete objects for all of the discretes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get discretes from

Return type

Array of Discrete objects

Example

To make an array of Discrete objects for all of the discretes in model m

```
var d = Discrete.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a discrete.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the discrete d:

```
var comm_array = d.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Discrete objects for all of the flagged discretes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get discretes from
flag	Flag	Flag set on the discretes that you want to retrieve

Return type

Array of Discrete objects

Example

To make an array of Discrete objects for all of the discretes in model m flagged with f

```
var d = Discrete.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Discrete object for a discrete ID.

Arguments

Name	Type	Description
Model	Model	Model to find the discrete in
number	integer	number of the discrete you want the Discrete object for

Return type

Discrete object (or null if discrete does not exist).

Example

To get the Discrete object for discrete 100 in model m

```
var d = Discrete.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Discrete property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Discrete.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	discrete property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Discrete property d.example is a parameter:

```
Options.property_parameter_names = true;
if (d.GetParameter(d.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Discrete property d.example is a parameter by using the GetParameter method:

```
if (d.ViewParameters().GetParameter(d.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this discrete (*ELEMENT_DISCRETE). **Note that a carriage return is not added**. See also Discrete.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for discrete m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the discrete. **Note that a carriage return is not added**. See also Discrete.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for discrete d:

```
var cards = d.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last discrete in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last discrete in

Return type

Discrete object (or null if there are no discretes in the model).

Example

To get the last discrete in model m:

```
var d = Discrete.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free discrete label in the model. Also see Discrete.FirstFreeLabel(), Discrete.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free discrete label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Discrete label.

Example

To get the last free discrete label in model m:

```
var label = Discrete.LastFreeLabel(m);
```

Next()

Description

Returns the next discrete in the model.

Arguments

No arguments

Return type

Discrete object (or null if there are no more discretes in the model).

Example

To get the discrete in model m after discrete d:

```
var d = d.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) discrete label in the model. Also see Discrete.FirstFreeLabel(), Discrete.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free discrete label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Discrete label.

Example

To get the next free discrete label in model m:

```
var label = Discrete.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a discrete.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only discretes from that model can be picked. If the argument is a Flag then only discretes that are flagged with *limit* can be selected. If omitted, or null, any discretes from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Discrete object (or null if not picked)

Example

To pick a discrete from model m giving the prompt 'Pick discrete from screen':

```
var d = Discrete.Pick('Pick discrete from screen', m);
```

Previous()

Description

Returns the previous discrete in the model.

Arguments

No arguments

Return type

Discrete object (or null if there are no more discretes in the model).

Example

To get the discrete in model m before discrete d:

```
var d = d.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the discretes in model m, from 1000000:

```
Discrete.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged discretes will be renumbered in
flag	Flag	Flag set on the discretes that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the discretes in model m flagged with f, from 1000000:

```
Discrete.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select discretes using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting discretes
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only discretes from that model can be selected. If the argument is a Flag then only discretes that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any discretes can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of discretes selected or null if menu cancelled

Example

To select discretes from model m, flagging those selected with flag f, giving the prompt 'Select discretes':

```
Discrete.Select(f, 'Select discretes', m);
```

To select discretes, flagging those selected with flag f but limiting selection to discretes flagged with flag l, giving the prompt 'Select discretes':

```
Discrete.Select(f, 'Select discretes', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the discrete.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the discrete

Return type

No return value

Example

To set flag f for discrete d:

```
d.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the discrete. The discrete will be sketched until you either call Discrete.Unsketch(), Discrete.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the discrete is sketched. If omitted redraw is true. If you want to sketch several discretes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch discrete d:

```
d.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged discretes in the model. The discretes will be sketched until you either call Discrete.Unsketch(), Discrete.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged discretes will be sketched in
flag	Flag	Flag set on the discretes that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the discretes are sketched. If omitted redraw is true. If you want to sketch flagged discretes several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all discretes flagged with flag in model m:

```
Discrete.SketchFlagged(m, flag);
```

Timestep()

Description

Calculates the timestep for the discrete

Arguments

No arguments

Return type

real

Example

To calculate the timestep for discrete d:

```
var timestep = d.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing discretes should be counted. If false or omitted referenced but undefined discretes will also be included in the total.

Return type

number of discretes

Example

To get the total number of discretes in model m:

```
var total = Discrete.Total(m);
```

Unblank()

Description

Unblanks the discrete

Arguments

No arguments

Return type

No return value

Example

To unblank discrete d:

```
d.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the discretes in model m:

```
Discrete.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged discretes will be unblanked in
flag	Flag	Flag set on the discretes that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the discretes in model m flagged with f:

```
Discrete.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all discretes will be unset in
flag	Flag	Flag to unset on the discretes

Return type

No return value

Example

To unset the flag f on all the discretes in model m:

```
Discrete.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the discrete.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the discrete is unsketched. If omitted redraw is true. If you want to unsketch several discretes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch discrete d:

```
d.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all discretes.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the discretes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all discretes in model m:

```
Discrete.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged discretes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discretes will be unsketched in
flag	Flag	Flag set on the discretes that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the discretes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all discretes flagged with flag in model m:

```
Discrete.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Discrete object.

Example

To check if Discrete property d.example is a parameter by using the Discrete.GetParameter() method:

```
if (d.ViewParameters().GetParameter(d.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for discrete. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for discrete d:

```
d.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this discrete.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for discrete d:

```
var xrefs = d.Xrefs();
```

---

toString()

Description

Creates a string containing the discrete data in keyword format. Note that this contains the keyword header and the keyword cards. See also Discrete.Keyword() and Discrete.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for discrete d in keyword format

```
var s = d.toString();
```

DiscreteSphere class

The DiscreteSphere class gives you access to element discrete sphere cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

DiscreteSphere properties

Name	Type	Description
colour	Colour	The colour of the discrete sphere
exists	logical	true if discrete sphere exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the discrete sphere is in.
inertia	real	Mass moment of inertia.
mass	real	Mass or volume value (depending on whether the _VOLUME option is set).
model	integer	The Model number that the discrete sphere is in.
nid	integer	Node ID.
pid	integer	Part ID to which this element belongs.
radius	real	Particle radius.
transparency	integer	The transparency of the discrete sphere (0-100) 0% is opaque, 100% is transparent.
volume	logical	Turns _VOLUME on or OFF. Note that this does NOT refer to the data field VOLUME. For the latter see the mass property.

Detailed Description

The DiscreteSphere class allows you to create, modify, edit and manipulate discrete sphere cards. See the documentation below for more details.

Constructor

new DiscreteSphere(Model*[Model]*, nid*[integer]*, pid*[integer]*, mass*[real]*, inertia*[real]*, radius*[real]*)

Description

Create a new DiscreteSphere object.

Arguments

Name	Type	Description
Model	Model	Model that discrete sphere will be created in
nid	integer	Node ID and Element ID are the same for discrete spheres.
pid	integer	Part ID to which this element belongs.
mass	real	Mass or volume value.
inertia	real	Mass moment of inertia.
radius	real	Particle radius.

Return type

DiscreteSphere object

Example

To create a new discrete sphere in model m with nid = 100, pid = 400, mass = 0.9, inertia = 2.5, radius = 2.0:

```
var dsph = new DiscreteSphere(m, 100, 400, 0.9, 2.5, 2.0);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a discrete sphere.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the discrete sphere

Return type

No return value

Example

To associate comment c to the discrete sphere dsph:

```
dsph.AssociateComment(c);
```

Blank()

Description

Blanks the discrete sphere

Arguments

No arguments

Return type

No return value

Example

To blank discrete sphere dsph:

```
dsph.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the discrete spheres in model m:

```
DiscreteSphere.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged discrete spheres will be blanked in
flag	Flag	Flag set on the discrete spheres that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the discrete spheres in model m flagged with f:

```
DiscreteSphere.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the discrete sphere is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if discrete sphere dsph is blanked:

```
if (dsph.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse discrete sphere dsph:

```
dsph.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the discrete sphere.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the discrete sphere

Return type

No return value

Example

To clear flag f for discrete sphere dsph:

```
dsph.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the discrete sphere.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

DiscreteSphere object

Example

To copy discrete sphere dsph into discrete sphere z:

```
var z = dsph.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a discrete sphere.

Arguments

Name	Type	Description
Model	Model	Model that the discrete sphere will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

DiscreteSphere object (or null if not made)

Example

To start creating a discrete sphere in model m:

```
var dsph = DiscreteSphere.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a discrete sphere.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the discrete sphere

Return type

No return value

Example

To detach comment c from the discrete sphere dsph:

```
dsph.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit discrete sphere dsph:

```
dsph.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for discrete sphere. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for discrete sphere dsph:

```
dsph.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for discrete sphere.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the discrete sphere colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the discrete sphere.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing discrete sphere dsph:

```
var colour = dsph.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first discrete sphere in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first discrete sphere in

Return type

DiscreteSphere object (or null if there are no discrete spheres in the model).

Example

To get the first discrete sphere in model m:

```
var dsph = DiscreteSphere.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free discrete sphere label in the model. Also see DiscreteSphere.LastFreeLabel(), DiscreteSphere.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free discrete sphere label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

DiscreteSphere label.

Example

To get the first free discrete sphere label in model m:

```
var label = DiscreteSphere.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the discrete spheres in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres will be flagged in
flag	Flag	Flag to set on the discrete spheres

Return type

No return value

Example

To flag all of the discrete spheres with flag f in model m:

```
DiscreteSphere.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the discrete sphere is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the discrete sphere

Return type

true if flagged, false if not.

Example

To check if discrete sphere dsph has flag f set on it:

```
if (dsph.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each discrete sphere in the model.
Note that ForEach has been designed to make looping over discrete spheres as fast as possible and so has some limitations.
Firstly, a single temporary DiscreteSphere object is created and on each function call it is updated with the current discrete sphere data. This means that you should not try to store the DiscreteSphere object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new discrete spheres inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres are in
func	function	Function to call for each discrete sphere
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the discrete spheres in model m:

```
DiscreteSphere.ForEach(m, test);
function test(dsph)
{
// dsph is DiscreteSphere object
}
```

To call function test for all of the discrete spheres in model m with optional object:

```
var data = { x:0, y:0 };
DiscreteSphere.ForEach(m, test, data);
function test(dsph, extra)
{
// dsph is DiscreteSphere object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of DiscreteSphere objects for all of the discrete spheres in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get discrete spheres from

Return type

Array of DiscreteSphere objects

Example

To make an array of DiscreteSphere objects for all of the discrete spheres in model m

```
var dsph = DiscreteSphere.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a discrete sphere.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the discrete sphere dsph:

```
var comm_array = dsph.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of DiscreteSphere objects for all of the flagged discrete spheres in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get discrete spheres from
flag	Flag	Flag set on the discrete spheres that you want to retrieve

Return type

Array of DiscreteSphere objects

Example

To make an array of DiscreteSphere objects for all of the discrete spheres in model m flagged with f

```
var dsph = DiscreteSphere.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the DiscreteSphere object for a discrete sphere ID.

Arguments

Name	Type	Description
Model	Model	Model to find the discrete sphere in
number	integer	number of the discrete sphere you want the DiscreteSphere object for

Return type

DiscreteSphere object (or null if discrete sphere does not exist).

Example

To get the DiscreteSphere object for discrete sphere 100 in model m

```
var dsph = DiscreteSphere.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a DiscreteSphere property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the DiscreteSphere.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	discrete sphere property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if DiscreteSphere property dsph.example is a parameter:

```
Options.property_parameter_names = true;
if (dsph.GetParameter(dsph.example) ) do_something...
Options.property_parameter_names = false;
```

To check if DiscreteSphere property dsph.example is a parameter by using the GetParameter method:

```
if (dsph.ViewParameters().GetParameter(dsph.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this discrete sphere (*ELEMENT_DISCRETE_SPHERE or *ELEMENT_DISCRETE_SPHERE_VOLUME). **Note that a carriage return is not added**. See also DiscreteSphere.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for discrete sphere dsph:

```
var key = dsph.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the discrete sphere. **Note that a carriage return is not added**. See also DiscreteSphere.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for discrete sphere dsph:

```
var cards = dsph.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last discrete sphere in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last discrete sphere in

Return type

DiscreteSphere object (or null if there are no discrete spheres in the model).

Example

To get the last discrete sphere in model m:

```
var dsph = DiscreteSphere.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free discrete sphere label in the model. Also see DiscreteSphere.FirstFreeLabel(), DiscreteSphere.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free discrete sphere label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

DiscreteSphere label.

Example

To get the last free discrete sphere label in model m:

```
var label = DiscreteSphere.LastFreeLabel(m);
```

Next()

Description

Returns the next discrete sphere in the model.

Arguments

No arguments

Return type

DiscreteSphere object (or null if there are no more discrete spheres in the model).

Example

To get the discrete sphere in model m after discrete sphere dsph:

```
var dsph = dsph.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) discrete sphere label in the model. Also see DiscreteSphere.FirstFreeLabel(), DiscreteSphere.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free discrete sphere label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

DiscreteSphere label.

Example

To get the next free discrete sphere label in model m:

```
var label = DiscreteSphere.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a discrete sphere.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only discrete spheres from that model can be picked. If the argument is a Flag then only discrete spheres that are flagged with *limit* can be selected. If omitted, or null, any discrete spheres from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

DiscreteSphere object (or null if not picked)

Example

To pick a discrete sphere from model m giving the prompt 'Pick discrete sphere from screen':

```
var dsph = DiscreteSphere.Pick('Pick discrete sphere from screen', m);
```

Previous()

Description

Returns the previous discrete sphere in the model.

Arguments

No arguments

Return type

DiscreteSphere object (or null if there are no more discrete spheres in the model).

Example

To get the discrete sphere in model m before discrete sphere dsph:

```
var dsph = dsph.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select discrete spheres using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting discrete spheres
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only discrete spheres from that model can be selected. If the argument is a Flag then only discrete spheres that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any discrete spheres can be selected. from any model.

modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of discrete spheres selected or null if menu cancelled

Example

To select discrete spheres from model m, flagging those selected with flag f, giving the prompt 'Select discrete spheres':

```
DiscreteSphere.Select(f, 'Select discrete spheres', m);
```

To select discrete spheres, flagging those selected with flag f but limiting selection to discrete spheres flagged with flag l, giving the prompt 'Select discrete spheres':

```
DiscreteSphere.Select(f, 'Select discrete spheres', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the discrete sphere.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the discrete sphere

Return type

No return value

Example

To set flag f for discrete sphere dsph:

```
dsph.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the discrete sphere. The discrete sphere will be sketched until you either call DiscreteSphere.Unsketch(), DiscreteSphere.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the discrete sphere is sketched. If omitted redraw is true. If you want to sketch several discrete spheres and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch discrete sphere dsph:

```
dsph.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged discrete spheres in the model. The discrete spheres will be sketched until you either call DiscreteSphere.Unsketch(), DiscreteSphere.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged discrete spheres will be sketched in
flag	Flag	Flag set on the discrete spheres that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the discrete spheres are sketched. If omitted redraw is true. If you want to sketch flagged discrete spheres several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all discrete spheres flagged with flag in model m:

```
DiscreteSphere.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing discrete spheres should be counted. If false or omitted referenced but undefined discrete spheres will also be included in the total.

Return type

number of discrete spheres

Example

To get the total number of discrete spheres in model m:

```
var total = DiscreteSphere.Total(m);
```

Unblank()

Description

Unblanks the discrete sphere

Arguments

No arguments

Return type

No return value

Example

To unblank discrete sphere dsph:

```
dsph.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the discrete spheres in model m:

```
DiscreteSphere.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged discrete spheres will be unblanked in
flag	Flag	Flag set on the discrete spheres that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the discrete spheres in model m flagged with f:

```
DiscreteSphere.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all discrete spheres will be unset in
flag	Flag	Flag to unset on the discrete spheres

Return type

No return value

Example

To unset the flag f on all the discrete spheres in model m:

```
DiscreteSphere.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the discrete sphere.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the discrete sphere is unsketched. If omitted redraw is true. If you want to unsketch several discrete spheres and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch discrete sphere dsph:

```
dsph.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all discrete spheres.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the discrete spheres are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all discrete spheres in model m:

```
DiscreteSphere.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged discrete spheres in the model.

Arguments

Name	Type	Description
Model	Model	Model that all discrete spheres will be unsketched in
flag	Flag	Flag set on the discrete spheres that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the discrete spheres are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all discrete spheres flagged with flag in model m:

```
DiscreteSphere.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

DiscreteSphere object.

Example

To check if DiscreteSphere property dsph.example is a parameter by using the DiscreteSphere.GetParameter() method:

```
if (dsph.ViewParameters().GetParameter(dsph.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for discrete sphere. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for discrete sphere dsph:

```
dsph.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this discrete sphere.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for discrete sphere dsph:

```
var xrefs = dsph.Xrefs();
```

---

toString()

Description

Creates a string containing the discrete sphere data in keyword format. Note that this contains the keyword header and the keyword cards. See also DiscreteSphere.Keyword() and DiscreteSphere.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for discrete sphere dsph in keyword format

```
var s = dsph.toString();
```

Mass class

The Mass class gives you access to element mass cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Mass constants

Name	Description
Mass.NODE_SET	Mass is *MASS_NODE_SET.

Mass properties

Name	Type	Description
colour	Colour	The colour of the mass
eid	integer	Mass number. Also see the label property which is an alternative name for this.
exists	logical	true if mass exists, false if referred to but not defined. (read only)
id	integer	Node id or node set id
include	integer	The Include file number that the mass is in.
label	integer	Mass number. Also see the eid property which is an alternative name for this.
mass	real	Mass value
model	integer	The Model number that the mass is in.
node_set	integer	The type of the mass. Can be false (*MASS) or Mass.NODE_SET (*MASS_NODE_SET)
pid	integer	Part ID
transparency	integer	The transparency of the mass (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Mass class allows you to create, modify, edit and manipulate mass cards. See the documentation below for more details.

Constructor

new Mass(Model*[Model]*, eid*[integer]*, id*[integer]*, mass*[real]*, node set (optional)*[integer]*)

Description

Create a new Mass object.

Arguments

Name	Type	Description
Model	Model	Model that mass will be created in
eid	integer	Mass number
id	integer	Node id or node set id
mass	real	Mass value
node set (optional)	integer	Only used if a node set is used

Return type

Mass object

Example

To create a new mass in model m with label 200, on node 500, or node set 500, with a mass of 3.5, use one of the following:

```
var m = new Mass(m, 200, 500, 3.5);

var m = new Mass(m, 200, 500, 3.5, Mass.NODE_SET);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a mass.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the mass

Return type

No return value

Example

To associate comment c to the mass m:

```
m.AssociateComment(c);
```

Blank()

Description

Blanks the mass

Arguments

No arguments

Return type

No return value

Example

To blank mass m:

```
m.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the masss in model m:

```
Mass.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged masss will be blanked in
flag	Flag	Flag set on the masss that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the masss in model m flagged with f:

```
Mass.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the mass is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if mass m is blanked:

```
if (m.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse mass m:

```
m.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the mass.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the mass

Return type

No return value

Example

To clear flag f for mass m:

```
m.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the mass.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Mass object

Example

To copy mass m into mass z:

```
var z = m.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a mass.

Arguments

Name	Type	Description
Model	Model	Model that the mass will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Mass object (or null if not made)

Example

To start creating a mass in model m:

```
var m = Mass.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a mass.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the mass

Return type

No return value

Example

To detach comment c from the mass m:

```
m.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit mass m:

```
m.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for mass. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for mass m:

```
m.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for mass.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the mass colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the mass.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing mass m:

```
var colour = m.ExtractColour();
```

First(Model[*Model*]) [static]

Description

Returns the first mass in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first mass in

Return type

Mass object (or null if there are no masss in the model).

Example

To get the first mass in model m:

```
var m = Mass.First(m);
```

FirstFreeLabel(Model[*Model*], layer (optional)[*Include number*]) [static]

Description

Returns the first free mass label in the model. Also see Mass.LastFreeLabel(), Mass.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free mass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Mass label.

Example

To get the first free mass label in model m:

```
var label = Mass.FirstFreeLabel(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the masss in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be flagged in
flag	Flag	Flag to set on the masss

Return type

No return value

Example

To flag all of the masss with flag f in model m:

```
Mass.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the mass is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the mass

Return type

true if flagged, false if not.

Example

To check if mass m has flag f set on it:

```
if (m.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each mass in the model.
Note that ForEach has been designed to make looping over masss as fast as possible and so has some limitations. Firstly, a single temporary Mass object is created and on each function call it is updated with the current mass data. This means that you should not try to store the Mass object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new masss inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all masss are in
func	function	Function to call for each mass
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the masss in model m:

```
Mass.ForEach(m, test);
function test(m)
{
// m is Mass object
}
```

To call function test for all of the masss in model m with optional object:

```
var data = { x:0, y:0 };
Mass.ForEach(m, test, data);
function test(m, extra)
{
// m is Mass object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Mass objects for all of the masss in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get masss from

Return type

Array of Mass objects

Example

To make an array of Mass objects for all of the masss in model m

```
var m = Mass.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a mass.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the mass m:

```
var comm_array = m.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Mass objects for all of the flagged masss in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get masss from
flag	Flag	Flag set on the masss that you want to retrieve

Return type

Array of Mass objects

Example

To make an array of Mass objects for all of the masss in model m flagged with f

```
var m = Mass.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Mass object for a mass ID.

Arguments

Name	Type	Description
Model	Model	Model to find the mass in
number	integer	number of the mass you want the Mass object for

Return type

Mass object (or null if mass does not exist).

Example

To get the Mass object for mass 100 in model m

```
var m = Mass.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Mass property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Mass.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	mass property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Mass property m.example is a parameter:

```
Options.property_parameter_names = true;
if (m.GetParameter(m.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Mass property m.example is a parameter by using the GetParameter method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this mass (*ELEMENT_MASS or *ELEMENT_MASS_NODE_SET). **Note that a carriage return is not added**. See also Mass.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for mass m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the mass. **Note that a carriage return is not added**. See also Mass.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for mass m:

```
var cards = m.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last mass in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last mass in

Return type

Mass object (or null if there are no masss in the model).

Example

To get the last mass in model m:

```
var m = Mass.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free mass label in the model. Also see Mass.FirstFreeLabel(), Mass.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free mass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Mass label.

Example

To get the last free mass label in model m:

```
var label = Mass.LastFreeLabel(m);
```

Next()

Description

Returns the next mass in the model.

Arguments

No arguments

Return type

Mass object (or null if there are no more masss in the model).

Example

To get the mass in model m after mass m:

```
var m = m.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) mass label in the model. Also see Mass.FirstFreeLabel(), Mass.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free mass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Mass label.

Example

To get the next free mass label in model m:

```
var label = Mass.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a mass.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only masss from that model can be picked. If the argument is a Flag then only masss that are flagged with *limit* can be selected. If omitted, or null, any masss from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Mass object (or null if not picked)

Example

To pick a mass from model m giving the prompt 'Pick mass from screen':

```
var m = Mass.Pick('Pick mass from screen', m);
```

Previous()

Description

Returns the previous mass in the model.

Arguments

No arguments

Return type

Mass object (or null if there are no more masss in the model).

Example

To get the mass in model m before mass m:

```
var m = m.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the masss in model m, from 1000000:

```
Mass.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged masss will be renumbered in
flag	Flag	Flag set on the masss that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the masss in model m flagged with f, from 1000000:

```
Mass.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select masss using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting masss
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only masss from that model can be selected. If the argument is a Flag then only masss that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any masss can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of masss selected or null if menu cancelled

Example

To select masss from model m, flagging those selected with flag f, giving the prompt 'Select masss':

```
Mass.Select(f, 'Select masss', m);
```

To select masss, flagging those selected with flag f but limiting selection to masss flagged with flag l, giving the prompt 'Select masss':

```
Mass.Select(f, 'Select masss', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the mass.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the mass

Return type

No return value

Example

To set flag f for mass m:

```
m.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the mass. The mass will be sketched until you either call Mass.Unsketch(), Mass.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the mass is sketched. If omitted redraw is true. If you want to sketch several masss and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch mass m:

```
m.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged masss in the model. The masss will be sketched until you either call Mass.Unsketch(), Mass.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged masss will be sketched in
flag	Flag	Flag set on the masss that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the masss are sketched. If omitted redraw is true. If you want to sketch flagged masss several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all masss flagged with flag in model m:

```
Mass.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of masss in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing masss should be counted. If false or omitted referenced but undefined masss will also be included in the total.

Return type

number of masss

Example

To get the total number of masss in model m:

```
var total = Mass.Total(m);
```

Unblank()

Description

Unblanks the mass

Arguments

No arguments

Return type

No return value

Example

To unblank mass m:

```
m.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the masss in model m:

```
Mass.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged masss will be unblanked in
flag	Flag	Flag set on the masss that you want to unblank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the masss in model m flagged with f:

```
Mass.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all masss will be unset in
flag	Flag	Flag to unset on the masss

Return type

No return value

Example

To unset the flag f on all the masss in model m:

```
Mass.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the mass.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the mass is unsketched. If omitted redraw is true. If you want to unsketch several masss and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch mass m:

```
m.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all masss.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the masss are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all masss in model m:

```
Mass.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged masss in the model.

Arguments

Name	Type	Description
Model	Model	Model that all masss will be unsketched in
flag	Flag	Flag set on the masss that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the masss are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all masss flagged with flag in model m:

```
Mass.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Mass object.

Example

To check if Mass property m.example is a parameter by using the Mass.GetParameter() method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for mass. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for mass m:

```
m.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this mass.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for mass m:

```
var xrefs = m.Xrefs();
```

---

toString()

Description

Creates a string containing the mass data in keyword format. Note that this contains the keyword header and the keyword cards. See also Mass.Keyword() and Mass.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for mass m in keyword format

```
var s = m.toString();
```

MassPart class

The MassPart class gives you access to element mass part cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- BlankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Create(Model *[Model]*, modal (optional) *[boolean]*)
- First(Model *[Model]*)
- FlagAll(Model *[Model]*, flag *[Flag]*)
- ForEach(Model *[Model]*, func *[function]*, extra (optional) *[any]*)
- GetAll(Model *[Model]*)
- GetFlagged(Model *[Model]*, flag *[Flag]*)
- GetFromID(Model *[Model]*, number *[integer]*)
- Last(Model *[Model]*)
- Pick(prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*, button text (optional) *[string]*)
- Select(flag *[Flag]*, prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*)
- SketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Total(Model *[Model]*, exists (optional) *[boolean]*)
- UnblankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnblankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- UnflagAll(Model *[Model]*, flag *[Flag]*)
- UnsketchAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnsketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)

Member functions

- AssociateComment(Comment *[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional) *[boolean]*)
- ClearFlag(flag *[Flag]*)
- Copy(range (optional) *[boolean]*)
- DetachComment(Comment *[Comment]*)
- Edit(modal (optional) *[boolean]*)
- Error(message *[string]*, details (optional) *[string]*)
- ExtractColour()
- Flagged(flag *[Flag]*)
- GetComments()
- GetParameter(prop *[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag *[Flag]*)
- Sketch(redraw (optional) *[boolean]*)
- Unblank()
- Unsketch(redraw (optional) *[boolean]*)
- ViewParameters()
- Warning(message *[string]*, details (optional) *[string]*)
- Xrefs()
- toString()

MassPart constants

Name	Description
MassPart.PART	Element is *ELEMENT_MASS_PART.
MassPart.SET	Element is *ELEMENT_MASS_PART_SET.

MassPart properties

Name	Type	Description
addmass	real	Added translational mass to be distributed to the nodes of the part or part set ID.
colour	Colour	The colour of the mass part
exists	logical	true if mass part exists, false if referred to but not defined. (read only)
finmass	real	Final translational mass of the part or part set ID.
id	integer	Part or part set ID if the SET option is active.
include	integer	The Include file number that the mass part is in.
lcid	integer	Optional load curve ID to scale the added mass at time = 0.
model	integer	The Model number that the element mass part is in.
mwd	integer	Optional flag for mass-weighted distribution.
option	constant	The Element Mass Part option. Can be MassPart.PART or MassPart.SET
transparency	integer	The transparency of the mass part (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The MassPart class allows you to create, modify, edit and manipulate element mass part cards. See the documentation below for more details.

Constructor

new MassPart(Model*[Model]*, option*[constant]*, id*[integer]*, addmass (optional)*[real]*, finmass (optional)*[real]*)

Description

Create a new MassPart object.

Arguments

Name	Type	Description
Model	Model	Model that mass part will be created in
option	constant	Suffix for element mass part. Can be MassPart.PART or MassPart.SET.
id	integer	Part or part set ID.
addmass (optional)	real	Added translational mass.
finmass (optional)	real	Final translational mass.

Return type

MassPart object

Example

To create a new element mass part in model m with option _<BLANK> and part ID 10:

```
var mp = new MassPart(m, MassPart.PART, 10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a element mass part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the element mass part

Return type

No return value

Example

To associate comment c to the element mass part mp:

```
mp.AssociateComment(c);
```

Blank()

Description

Blanks the element mass part

Arguments

No arguments

Return type

No return value

Example

To blank element mass part mp:

```
mp.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the element mass parts in model m:

```
MassPart.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged element mass parts will be blanked in
flag	Flag	Flag set on the element mass parts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the element mass parts in model m flagged with f:

```
MassPart.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the element mass part is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if element mass part mp is blanked:

```
if (mp.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse element mass part mp:

```
mp.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the element mass part.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the element mass part

Return type

No return value

Example

To clear flag f for element mass part mp:

```
mp.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the element mass part.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

MassPart object

Example

To copy element mass part mp into element mass part z:

```
var z = mp.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a mass part.

Arguments

Name	Type	Description
Model	Model	Model that the mass part will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

MassPart object (or null if not made)

Example

To start creating a mass part in model m:

```
var mp = MassPart.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a element mass part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the element mass part

Return type

No return value

Example

To detach comment c from the element mass part mp:

```
mp.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit element mass part mp:

```
mp.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for element mass part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for element mass part mp:

```
mp.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for element mass part.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the element mass part colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the element mass part.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing element mass part mp:

```
var colour = mp.ExtractColour();
```

First(Model[*Model*]) [static]

Description

Returns the first element mass part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first element mass part in

Return type

MassPart object (or null if there are no element mass parts in the model).

Example

To get the first element mass part in model m:

```
var mp = MassPart.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the element mass parts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts will be flagged in
flag	Flag	Flag to set on the element mass parts

Return type

No return value

Example

To flag all of the element mass parts with flag f in model m:

```
MassPart.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the element mass part is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the element mass part

Return type

true if flagged, false if not.

Example

To check if element mass part mp has flag f set on it:

```
if (mp.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each element mass part in the model.
Note that ForEach has been designed to make looping over element mass parts as fast as possible and so has some limitations.
Firstly, a single temporary MassPart object is created and on each function call it is updated with the current element mass part data. This means that you should not try to store the MassPart object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new element mass parts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts are in
func	function	Function to call for each element mass part
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the element mass parts in model m:

```
MassPart.ForEach(m, test);
function test(mp)
{
// mp is MassPart object
}
```

To call function test for all of the element mass parts in model m with optional object:

```
var data = { x:0, y:0 };
MassPart.ForEach(m, test, data);
function test(mp, extra)
{
// mp is MassPart object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of MassPart objects for all of the element mass parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get element mass parts from

Return type

Array of MassPart objects

Example

To make an array of MassPart objects for all of the element mass parts in model m

```
var mp = MassPart.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a element mass part.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the element mass part mp:

```
var comm_array = mp.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of MassPart objects for all of the flagged element mass parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get element mass parts from
flag	Flag	Flag set on the element mass parts that you want to retrieve

Return type

Array of MassPart objects

Example

To make an array of MassPart objects for all of the element mass parts in model m flagged with f

```
var mp = MassPart.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the MassPart object for a element mass part ID.

Arguments

Name	Type	Description

Model	Model	Model to find the element mass part in
number	integer	number of the element mass part you want the MassPart object for

Return type

MassPart object (or null if element mass part does not exist).

Example

To get the MassPart object for element mass part 100 in model m

```
var mp = MassPart.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a MassPart property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the MassPart.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	element mass part property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if MassPart property mp.example is a parameter:

```
Options.property_parameter_names = true;
if (mp.GetParameter(mp.example) ) do_something...
Options.property_parameter_names = false;
```

To check if MassPart property mp.example is a parameter by using the GetParameter method:

```
if (mp.ViewParameters().GetParameter(mp.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this mass part (*ELEMENT_MASS_PART) **Note that a carriage return is not added**. See also MassPart.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for mass part mp:

```
var key = mp.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the mass part. **Note that a carriage return is not added**. See also MassPart.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for mass part mp:

```
var cards = mp.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last element mass part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last element mass part in

Return type

MassPart object (or null if there are no element mass parts in the model).

Example

To get the last element mass part in model m:

```
var mp = MassPart.Last(m);
```

Next()

Description

Returns the next element mass part in the model.

Arguments

No arguments

Return type

MassPart object (or null if there are no more element mass parts in the model).

Example

To get the element mass part in model m after element mass part mp:

```
var mp = mp.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a element mass part.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only element mass parts from that model can be picked. If the argument is a Flag then only element mass parts that are flagged with *limit* can be selected. If omitted, or null, any element mass parts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

MassPart object (or null if not picked)

Example

To pick a element mass part from model m giving the prompt 'Pick element mass part from screen':

```
var mp = MassPart.Pick('Pick element mass part from screen', m);
```

Previous()

Description

Returns the previous element mass part in the model.

Arguments

No arguments

Return type

MassPart object (or null if there are no more element mass parts in the model).

Example

To get the element mass part in model m before element mass part mp:

```
var mp = mp.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select element mass parts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting element mass parts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only element mass parts from that model can be selected. If the argument is a Flag then only element mass parts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any element mass parts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of element mass parts selected or null if menu cancelled

Example

To select element mass parts from model m, flagging those selected with flag f, giving the prompt 'Select element mass parts':

```
MassPart.Select(f, 'Select element mass parts', m);
```

To select element mass parts, flagging those selected with flag f but limiting selection to element mass parts flagged with flag l, giving the prompt 'Select element mass parts':

```
MassPart.Select(f, 'Select element mass parts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the element mass part.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the element mass part

Return type

No return value

Example

To set flag f for element mass part mp:

```
mp.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the element mass part. The element mass part will be sketched until you either call MassPart.Unsketch(), MassPart.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the element mass part is sketched. If omitted redraw is true. If you want to sketch several element mass parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch element mass part mp:

```
mp.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged element mass parts in the model. The element mass parts will be sketched until you either call MassPart.Unsketch(), MassPart.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged element mass parts will be sketched in
flag	Flag	Flag set on the element mass parts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the element mass parts are sketched. If omitted redraw is true. If you want to sketch flagged element mass parts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all element mass parts flagged with flag in model m:

```
MassPart.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing element mass parts should be counted. If false or omitted referenced but undefined element mass parts will also be included in the total.

Return type

number of element mass parts

Example

To get the total number of element mass parts in model m:

```
var total = MassPart.Total(m);
```

Unblank()

Description

Unblanks the element mass part

Arguments

No arguments

Return type

No return value

Example

To unblank element mass part mp:

```
mp.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the element mass parts in model m:

```
MassPart.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged element mass parts will be unblanked in
flag	Flag	Flag set on the element mass parts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the element mass parts in model m flagged with f:

```
MassPart.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all element mass parts will be unset in
flag	Flag	Flag to unset on the element mass parts

Return type

No return value

Example

To unset the flag f on all the element mass parts in model m:

```
MassPart.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the element mass part.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the element mass part is unsketched. If omitted redraw is true. If you want to unsketch several element mass parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch element mass part mp:

```
mp.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all element mass parts.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the element mass parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all element mass parts in model m:

```
MassPart.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged element mass parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all element mass parts will be unsketched in
flag	Flag	Flag set on the element mass parts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the element mass parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all element mass parts flagged with flag in model m:

```
MassPart.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

MassPart object.

Example

To check if MassPart property mp.example is a parameter by using the MassPart.GetParameter() method:

```
if (mp.ViewParameters().GetParameter(mp.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for element mass part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for element mass part mp:

```
mp.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this element mass part.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for element mass part mp:

```
var xrefs = mp.Xrefs();
```

---

toString()

Description

Creates a string containing the mass part data in keyword format. Note that this contains the keyword header and the keyword cards. See also MassPart.Keyword() and MassPart.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for mass part mp in keyword format

```
var str = mp.toString();
```

Pretensioner class

The Pretensioner class gives you access to seatbelt pretensioner cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Pretensioner properties

Name	Type	Description
colour	Colour	The colour of the pretensioner
exists	logical	true if pretensioner exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the pretensioner is in.
label	integer	Pretensioner number. Also see the sbprid property which is an alternative name for this.
lmtfrc	real	Limiting force
model	integer	The Model number that the pretensioner is in.
ptlcid	integer	Loadcurve of pull-in vs time
sbprid	integer	Pretensioner number. Also see the label property which is an alternative name for this.
sbprty	integer	Pretensioner type.
sbrid	integer	Retractor number.
sbsid1	integer	Sensor number 1
sbsid2	integer	Sensor number 2
sbsid3	integer	Sensor number 3
sbsid4	integer	Sensor number 4
time	real	Time between sensor triggering and pretensioner acting.
transparency	integer	The transparency of the pretensioner (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Pretensioner class allows you to create, modify, edit and manipulate seatbelt pretensioner cards. See the documentation below for more details.

Constructor

new Pretensioner(Model*[Model]*, sbprid*[integer]*, sbprty*[integer]*, sbrid*[integer]*, ptlcid*[integer]*, sbsid1*[integer]*, sbsid2 (optional)*[integer]*, sbsid3 (optional)*[integer]*, sbsid4 (optional)*[integer]*, time (optional)*[real]*, lmtfrc (optional)*[real]*)

Description

Create a new Seatbelt Pretensioner object.

Arguments

Name	Type	Description
Model	Model	Model that pretensioner will be created in
sbprid	integer	Pretensioner number.
sbprty	integer	Pretensioner type.
sbrid	integer	Retractor number.
ptlcid	integer	Loadcurve of pull-in vs time
sbsid1	integer	Sensor number 1

sbsid2 (optional)	integer	Sensor number 2
sbsid3 (optional)	integer	Sensor number 3
sbsid4 (optional)	integer	Sensor number 4
time (optional)	real	Time between sensor triggering and pretensioner acting.
lmtfrc (optional)	real	Limiting force

Return type

Pretensioner object

Example

To create a new pyrotechnic seatbelt pretensioner in model m with label 100, Retractor 10, Loading curve 20 and Sensor 30:

```
var p = new Pretensioner(m, 100, 1, 10, 20, 30);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a pretensioner.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the pretensioner

Return type

No return value

Example

To associate comment c to the pretensioner p:

```
p.AssociateComment(c);
```

Blank()

Description

Blanks the pretensioner

Arguments

No arguments

Return type

No return value

Example

To blank pretensioner p:

```
p.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the pretensioners in model m:

```
Pretensioner.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged pretensioners will be blanked in
flag	Flag	Flag set on the pretensioners that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the pretensioners in model m flagged with f:

```
Pretensioner.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the pretensioner is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if pretensioner p is blanked:

```
if (p.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse pretensioner p:

```
p.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the pretensioner.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the pretensioner

Return type

No return value

Example

To clear flag f for pretensioner p:

```
p.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the pretensioner.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Pretensioner object

Example

To copy pretensioner p into pretensioner z:

```
var z = p.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a pretensioner.

Arguments

Name	Type	Description
Model	Model	Model that the pretensioner will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Pretensioner object (or null if not made)

Example

To start creating an pretensioner in model m:

```
var p = Pretensioner.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a pretensioner.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the pretensioner

Return type

No return value

Example

To detach comment c from the pretensioner p:

```
p.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit pretensioner p:

```
p.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for pretensioner. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for pretensioner p:

```
p.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for pretensioner.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the pretensioner colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the pretensioner.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing pretensioner p:

```
var colour = p.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first pretensioner in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first pretensioner in

Return type

Pretensioner object (or null if there are no pretensioners in the model).

Example

To get the first pretensioner in model m:

```
var p = Pretensioner.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free pretensioner label in the model. Also see Pretensioner.LastFreeLabel(), Pretensioner.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free pretensioner label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Pretensioner label.

Example

To get the first free pretensioner label in model m:

```
var label = Pretensioner.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the pretensioners in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be flagged in
flag	Flag	Flag to set on the pretensioners

Return type

No return value

Example

To flag all of the pretensioners with flag f in model m:

```
Pretensioner.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the pretensioner is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the pretensioner

Return type

true if flagged, false if not.

Example

To check if pretensioner p has flag f set on it:

```
if (p.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each pretensioner in the model.
Note that ForEach has been designed to make looping over pretensioners as fast as possible and so has some limitations.
Firstly, a single temporary Pretensioner object is created and on each function call it is updated with the current pretensioner data. This means that you should not try to store the Pretensioner object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new pretensioners inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners are in
func	function	Function to call for each pretensioner
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the pretensioners in model m:

```
Pretensioner.ForEach(m, test);
function test(p)
{
// p is Pretensioner object
}
```

To call function test for all of the pretensioners in model m with optional object:

```
var data = { x:0, y:0 };
Pretensioner.ForEach(m, test, data);
function test(p, extra)
{
// p is Pretensioner object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Pretensioner objects for all of the pretensioners in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get pretensioners from

Return type

Array of Pretensioner objects

Example

To make an array of Pretensioner objects for all of the pretensioners in model m

```
var p = Pretensioner.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a pretensioner.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the pretensioner p:

```
var comm_array = p.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Pretensioner objects for all of the flagged pretensioners in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get pretensioners from
flag	Flag	Flag set on the pretensioners that you want to retrieve

Return type

Array of Pretensioner objects

Example

To make an array of Pretensioner objects for all of the pretensioners in model m flagged with f

```
var p = Pretensioner.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Pretensioner object for a pretensioner ID.

Arguments

Name	Type	Description
Model	Model	Model to find the pretensioner in
number	integer	number of the pretensioner you want the Pretensioner object for

Return type

Pretensioner object (or null if pretensioner does not exist).

Example

To get the Pretensioner object for pretensioner 100 in model m

```
var p = Pretensioner.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Pretensioner property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Pretensioner.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	pretensioner property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Pretensioner property p.example is a parameter:

```
Options.property_parameter_names = true;
if (p.GetParameter(p.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Pretensioner property p.example is a parameter by using the GetParameter method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this pretensioner (*ELEMENT_SEATBELT_PRETEROMETER) **Note that a carriage return is not added**. See also Pretensioner.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for pretensioner p:

```
var key = p.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the pretensioner. **Note that a carriage return is not added**. See also Pretensioner.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for pretensioner a:

```
var cards = a.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last pretensioner in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last pretensioner in

Return type

Pretensioner object (or null if there are no pretensioners in the model).

Example

To get the last pretensioner in model m:

```
var p = Pretensioner.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free pretensioner label in the model. Also see Pretensioner.FirstFreeLabel(), Pretensioner.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free pretensioner label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Pretensioner label.

Example

To get the last free pretensioner label in model m:

```
var label = Pretensioner.LastFreeLabel(m);
```

Next()

Description

Returns the next pretensioner in the model.

Arguments

No arguments

Return type

Pretensioner object (or null if there are no more pretensioners in the model).

Example

To get the pretensioner in model m after pretensioner p:

```
var p = p.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) pretensioner label in the model. Also see Pretensioner.FirstFreeLabel(), Pretensioner.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free pretensioner label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Pretensioner label.

Example

To get the next free pretensioner label in model m:

```
var label = Pretensioner.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a pretensioner.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only pretensioners from that model can be picked. If the argument is a Flag then only pretensioners that are flagged with *limit* can be selected. If omitted, or null, any pretensioners from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Pretensioner object (or null if not picked)

Example

To pick a pretensioner from model m giving the prompt 'Pick pretensioner from screen':

```
var p = Pretensioner.Pick('Pick pretensioner from screen', m);
```

Previous()

Description

Returns the previous pretensioner in the model.

Arguments

No arguments

Return type

Pretensioner object (or null if there are no more pretensioners in the model).

Example

To get the pretensioner in model m before pretensioner p:

```
var p = p.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the pretensioners in model m, from 1000000:

```
Pretensioner.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged pretensioners will be renumbered in
flag	Flag	Flag set on the pretensioners that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the pretensioners in model m flagged with f, from 1000000:

```
Pretensioner.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select pretensioners using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting pretensioners
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only pretensioners from that model can be selected. If the argument is a Flag then only pretensioners that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any pretensioners can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of pretensioners selected or null if menu cancelled

Example

To select pretensioners from model m, flagging those selected with flag f, giving the prompt 'Select pretensioners':

```
Pretensioner.Select(f, 'Select pretensioners', m);
```

To select pretensioners, flagging those selected with flag f but limiting selection to pretensioners flagged with flag l, giving the prompt 'Select pretensioners':

```
Pretensioner.Select(f, 'Select pretensioners', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the pretensioner.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the pretensioner

Return type

No return value

Example

To set flag f for pretensioner p:

```
p.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the pretensioner. The pretensioner will be sketched until you either call Pretensioner.Unsketch(), Pretensioner.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the pretensioner is sketched. If omitted redraw is true. If you want to sketch several pretensioners and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch pretensioner p:

```
p.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged pretensioners in the model. The pretensioners will be sketched until you either call Pretensioner.Unsketch(), Pretensioner.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged pretensioners will be sketched in
flag	Flag	Flag set on the pretensioners that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the pretensioners are sketched. If omitted redraw is true. If you want to sketch flagged pretensioners several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all pretensioners flagged with flag in model m:

```
Pretensioner.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing pretensioners should be counted. If false or omitted referenced but undefined pretensioners will also be included in the total.

Return type

number of pretensioners

Example

To get the total number of pretensioners in model m:

```
var total = Pretensioner.Total(m);
```

Unblank()

Description

Unblanks the pretensioner

Arguments

No arguments

Return type

No return value

Example

To unblank pretensioner p:

```
p.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the pretensioners in model m:

```
Pretensioner.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged pretensioners will be unblanked in

flag	Flag	Flag set on the pretensioners that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the pretensioners in model m flagged with f:

```
Pretensioner.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all pretensioners will be unset in
flag	Flag	Flag to unset on the pretensioners

Return type

No return value

Example

To unset the flag f on all the pretensioners in model m:

```
Pretensioner.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the pretensioner.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the pretensioner is unsketched. If omitted redraw is true. If you want to unsketch several pretensioners and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch pretensioner p:

```
p.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all pretensioners.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the pretensioners are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all pretensioners in model m:

```
Pretensioner.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged pretensioners in the model.

Arguments

Name	Type	Description
Model	Model	Model that all pretensioners will be unsketched in
flag	Flag	Flag set on the pretensioners that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the pretensioners are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all pretensioners flagged with flag in model m:

```
Pretensioner.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Pretensioner object.

Example

To check if Pretensioner property p.example is a parameter by using the Pretensioner.GetParameter() method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for pretensioner. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for pretensioner p:

```
p.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this pretensioner.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for pretensioner p:

```
var xrefs = p.Xrefs();
```

---

toString()

Description

Creates a string containing the pretensioner data in keyword format. Note that this contains the keyword header and the keyword cards. See also Pretensioner.Keyword() and Pretensioner.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for pretensioner p in keyword format

```
var str = p.toString();
```

Retractor class

The Retractor class gives you access to seatbelt retractor cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Retractor properties

Name	Type	Description
colour	Colour	The colour of the retractor
dsid	integer	Retractor deactivation Sensor
exists	logical	true if retractor exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the retractor is in.
label	integer	Retractor number. Also see the sbrid property which is an alternative name for this.
lfed	real	Fed length
llcid	integer	Loadcurve for loading (pull-out vs force)
model	integer	The Model number that the retractor is in.
pull	real	Amount of pull out between time delay ending and retractor locking
sbid	integer	Seatbelt1D number (or Set Shell number if sbrnid is negative).
sbrid	integer	Retractor number. Also see the label property which is an alternative name for this.
sbrnid	integer	Node number (or Set Node number if negative).
shell_seatbelt	logical	true if retractor is used for shell (2D) seatbelt elements. (read only)
sid1	integer	Sensor number 1
sid2	integer	Sensor number 2
sid3	integer	Sensor number 3
sid4	integer	Sensor number 4
tdel	real	Time delay after sensor triggers
transparency	integer	The transparency of the retractor (0-100) 0% is opaque, 100% is transparent.
ulcid	integer	Loadcurve for unloading (pull-out vs force)

Detailed Description

The Retractor class allows you to create, modify, edit and manipulate seatbelt retractor cards. See the documentation below for more details.

Constructor

new Retractor(Model*[Model]*, sbrid*[integer]*, sbrnid*[integer]*, sbid*[integer]*, llcid*[integer]*, sid1*[integer]*, sid2 (optional)*[integer]*, sid3 (optional)*[integer]*, sid4 (optional)*[integer]*, tdel (optional)*[real]*, pull (optional)*[real]*, ulcid (optional)*[integer]*, lfed (optional)*[real]*)

Description

Create a new Seatbelt Retractor object.

Arguments

Name	Type	Description
Model	Model	Model that retractor will be created in
sbrid	integer	Retractor number.

sbrnid	integer	Node number (or Set Node number if negative).
sbid	integer	Seatbelt number. (or Set Shell number if sbrnid is negative)
llcid	integer	Loadcurve for loading (pull-out vs force)
sid1	integer	Sensor number 1
sid2 (optional)	integer	Sensor number 2
sid3 (optional)	integer	Sensor number 3
sid4 (optional)	integer	Sensor number 4
tdel (optional)	real	Time delay after sensor triggers.
pull (optional)	real	Amount of pull out between time delay ending and retractor locking.
ulcid (optional)	integer	Loadcurve for unloading (pull-out vs force)
lfed (optional)	real	Fed length

Return type

Retractor object

Example

To create a new seatbelt retractor in model m with label 100, retractor Node 10, Seatbelt 20, Loading curve 30 and Sensor 40:

```
var a = new Retractor(m, 100, 10, 20, 30, 40);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a retractor.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the retractor

Return type

No return value

Example

To associate comment c to the retractor r:

```
r.AssociateComment(c);
```

Blank()

Description

Blanks the retractor

Arguments

No arguments

Return type

No return value

Example

To blank retractor r:

```
r.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the retractors in model m:

```
Retractor.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged retractors will be blanked in
flag	Flag	Flag set on the retractors that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the retractors in model m flagged with f:

```
Retractor.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the retractor is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if retractor r is blanked:

```
if (r.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse retractor r:

```
r.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the retractor.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the retractor

Return type

No return value

Example

To clear flag f for retractor r:

```
r.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the retractor.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Retractor object

Example

To copy retractor r into retractor z:

```
var z = r.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a retractor.

Arguments

Name	Type	Description
Model	Model	Model that the retractor will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Retractor object (or null if not made)

Example

To start creating an retractor in model m:

```
var r = Retractor.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a retractor.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the retractor

Return type

No return value

Example

To detach comment c from the retractor r:

```
r.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit retractor r:

```
r.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for retractor. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for retractor r:

```
r.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for retractor.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the retractor colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the retractor.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing retractor r:

```
var colour = r.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first retractor in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first retractor in

Return type

Retractor object (or null if there are no retractors in the model).

Example

To get the first retractor in model m:

```
var r = Retractor.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free retractor label in the model. Also see Retractor.LastFreeLabel(), Retractor.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free retractor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Retractor label.

Example

To get the first free retractor label in model m:

```
var label = Retractor.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the retractors in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be flagged in
flag	Flag	Flag to set on the retractors

Return type

No return value

Example

To flag all of the retractors with flag f in model m:

```
Retractor.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the retractor is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the retractor

Return type

true if flagged, false if not.

Example

To check if retractor r has flag f set on it:

```
if (r.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each retractor in the model.
Note that ForEach has been designed to make looping over retractors as fast as possible and so has some limitations.
Firstly, a single temporary Retractor object is created and on each function call it is updated with the current retractor data. This means that you should not try to store the Retractor object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new retractors inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all retractors are in
func	function	Function to call for each retractor
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the retractors in model m:

```
Retractor.ForEach(m, test);
function test(r)
{
// r is Retractor object
}
```

To call function test for all of the retractors in model m with optional object:

```
var data = { x:0, y:0 };
Retractor.ForEach(m, test, data);
function test(r, extra)
{
// r is Retractor object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Retractor objects for all of the retractors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get retractors from

Return type

Array of Retractor objects

Example

To make an array of Retractor objects for all of the retractors in model m

```
var r = Retractor.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a retractor.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the retractor r:

```
var comm_array = r.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Retractor objects for all of the flagged retractors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get retractors from
flag	Flag	Flag set on the retractors that you want to retrieve

Return type

Array of Retractor objects

Example

To make an array of Retractor objects for all of the retractors in model m flagged with f

```
var r = Retractor.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Retractor object for a retractor ID.

Arguments

Name	Type	Description
Model	Model	Model to find the retractor in
number	integer	number of the retractor you want the Retractor object for

Return type

Retractor object (or null if retractor does not exist).

Example

To get the Retractor object for retractor 100 in model m

```
var r = Retractor.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Retractor property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Retractor.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	retractor property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Retractor property r.example is a parameter:

```
Options.property_parameter_names = true;
if (r.GetParameter(r.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Retractor property r.example is a parameter by using the GetParameter method:

```
if (r.ViewParameters().GetParameter(r.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this retractor (*ELEMENT_SEATBELT_RETREROMETER) **Note that a carriage return is not added**. See also Retractor.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for retractor r:

```
var key = r.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the retractor. **Note that a carriage return is not added**. See also Retractor.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for retractor r:

```
var cards = r.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last retractor in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last retractor in

Return type

Retractor object (or null if there are no retractors in the model).

Example

To get the last retractor in model m:

```
var r = Retractor.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free retractor label in the model. Also see Retractor.FirstFreeLabel(), Retractor.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free retractor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Retractor label.

Example

To get the last free retractor label in model m:

```
var label = Retractor.LastFreeLabel(m);
```

Next()

Description

Returns the next retractor in the model.

Arguments

No arguments

Return type

Retractor object (or null if there are no more retractors in the model).

Example

To get the retractor in model m after retractor r:

```
var r = r.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) retractor label in the model. Also see Retractor.FirstFreeLabel(), Retractor.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free retractor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Retractor label.

Example

To get the next free retractor label in model m:

```
var label = Retractor.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a retractor.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only retractors from that model can be picked. If the argument is a Flag then only retractors that are flagged with *limit* can be selected. If omitted, or null, any retractors from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Retractor object (or null if not picked)

Example

To pick a retractor from model m giving the prompt 'Pick retractor from screen':

```
var r = Retractor.Pick('Pick retractor from screen', m);
```

Previous()

Description

Returns the previous retractor in the model.

Arguments

No arguments

Return type

Retractor object (or null if there are no more retractors in the model).

Example

To get the retractor in model m before retractor r:

```
var r = r.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the retractors in model m, from 1000000:

```
Retractor.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged retractors will be renumbered in
flag	Flag	Flag set on the retractors that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the retractors in model m flagged with f, from 1000000:

```
Retractor.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select retractors using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting retractors
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only retractors from that model can be selected. If the argument is a Flag then only retractors that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any retractors can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of retractors selected or null if menu cancelled

Example

To select retractors from model m, flagging those selected with flag f, giving the prompt 'Select retractors':

```
Retractor.Select(f, 'Select retractors', m);
```

To select retractors, flagging those selected with flag f but limiting selection to retractors flagged with flag l, giving the prompt 'Select retractors':

```
Retractor.Select(f, 'Select retractors', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the retractor.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the retractor

Return type

No return value

Example

To set flag f for retractor r:

```
r.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the retractor. The retractor will be sketched until you either call Retractor.Unsketch(), Retractor.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the retractor is sketched. If omitted redraw is true. If you want to sketch several retractors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch retractor r:

```
r.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged retractors in the model. The retractors will be sketched until you either call Retractor.Unsketch(), Retractor.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged retractors will be sketched in
flag	Flag	Flag set on the retractors that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the retractors are sketched. If omitted redraw is true. If you want to sketch flagged retractors several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all retractors flagged with flag in model m:

```
Retractor.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing retractors should be counted. If false or omitted referenced but undefined retractors will also be included in the total.

Return type

number of retractors

Example

To get the total number of retractors in model m:

```
var total = Retractor.Total(m);
```

Unblank()

Description

Unblanks the retractor

Arguments

No arguments

Return type

No return value

Example

To unblank retractor r:

```
r.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the retractors in model m:

```
Retractor.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged retractors will be unblanked in
flag	Flag	Flag set on the retractors that you want to unblank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the retractors in model m flagged with f:

```
Retractor.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all retractors will be unset in
flag	Flag	Flag to unset on the retractors

Return type

No return value

Example

To unset the flag f on all the retractors in model m:

```
Retractor.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the retractor.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the retractor is unsketched. If omitted redraw is true. If you want to unsketch several retractors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch retractor r:

```
r.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all retractors.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the retractors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all retractors in model m:

```
Retractor.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged retractors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all retractors will be unsketched in
flag	Flag	Flag set on the retractors that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the retractors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all retractors flagged with flag in model m:

```
Retractor.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Retractor object.

Example

To check if Retractor property r.example is a parameter by using the Retractor.GetParameter() method:

```
if (r.ViewParameters().GetParameter(r.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for retractor. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for retractor r:

```
r.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this retractor.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for retractor r:

```
var xrefs = r.Xrefs();
```

---

toString()

Description

Creates a string containing the retractor data in keyword format. Note that this contains the keyword header and the keyword cards. See also Retractor.Keyword() and Retractor.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for retractor r in keyword format

```
var str = r.toString();
```

Seatbelt1D class

The Seatbelt1D class gives you access to 2 noded (1D) element seatbelt cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)

- Xrefs()
- toString()

Seatbelt1D properties

Name	Type	Description
colour	Colour	The colour of the seatbelt
eid	integer	Seatbelt1D number. Also see the label property which is an alternative name for this.
exists	logical	true if seatbelt exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the seatbelt is in.
label	integer	Seatbelt1D number. Also see the eid property which is an alternative name for this.
model	integer	The Model number that the seatbelt is in.
n1	integer	Node 1 ID
n2	integer	Node 2 ID
pid	integer	Part ID
sbrid	integer	Retractor ID
slen	real	Initial slack length
transparency	integer	The transparency of the seatbelt (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Seatbelt1D class allows you to create, modify, edit and manipulate 2 noded (1D) element seatbelt cards. See the documentation below for more details.

Constructor

new Seatbelt1D(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*)

Description

Create a new Seatbelt1D object.

Arguments

Name	Type	Description
Model	Model	Model that seatbelt will be created in
eid	integer	Seatbelt ID.
pid	integer	Part number.
n1	integer	Node 1 ID
n2	integer	Node 2 ID

Return type

Seatbelt1D object

Example

To create a new 2 noded element seatbelt in model m with label 100, part 10 and nodes 20, 21:

```
var a = new Seatbelt1D(m, 100, 10, 20, 21);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a seatbelt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the seatbelt

Return type

No return value

Example

To associate comment c to the seatbelt s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the seatbelt

Arguments

No arguments

Return type

No return value

Example

To blank seatbelt s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the seatbelts in model m:

```
Seatbelt1D.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged seatbelts will be blanked in
flag	Flag	Flag set on the seatbelts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the seatbelts in model m flagged with f:

```
Seatbelt1D.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the seatbelt is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if seatbelt s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse seatbelt s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the seatbelt.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the seatbelt

Return type

No return value

Example

To clear flag f for seatbelt s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the seatbelt.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Seatbelt1D object

Example

To copy seatbelt s into seatbelt z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a 2 noded seatbelt.

Arguments

Name	Type	Description
Model	Model	Model that the seatbelt will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Seatbelt1D object (or null if not made)

Example

To start creating a seatbelt in model m:

```
var s = Seatbelt1D.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a seatbelt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the seatbelt

Return type

No return value

Example

To detach comment c from the seatbelt s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit seatbelt s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for seatbelt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for seatbelt s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for seatbelt.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the seatbelt colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the seatbelt.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing seatbelt s:

```
var colour = s.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first seatbelt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first seatbelt in

Return type

Seatbelt1D object (or null if there are no seatbelts in the model).

Example

To get the first seatbelt in model m:

```
var s = Seatbelt1D.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free seatbelt label in the model. Also see Seatbelt1D.LastFreeLabel(), Seatbelt1D.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Seatbelt1D label.

Example

To get the first free seatbelt label in model m:

```
var label = Seatbelt1D.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the seatbelts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be flagged in
flag	Flag	Flag to set on the seatbelts

Return type

No return value

Example

To flag all of the seatbelts with flag f in model m:

```
Seatbelt1D.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the seatbelt is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the seatbelt

Return type

true if flagged, false if not.

Example

To check if seatbelt s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each seatbelt in the model.
Note that ForEach has been designed to make looping over seatbelts as fast as possible and so has some limitations.
Firstly, a single temporary Seatbelt1D object is created and on each function call it is updated with the current seatbelt data. This means that you should not try to store the Seatbelt1D object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new seatbelts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts are in
func	function	Function to call for each seatbelt
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the seatbelts in model m:

```
Seatbelt1D.ForEach(m, test);
function test(s)
{
// s is Seatbelt1D object
}
```

To call function test for all of the seatbelts in model m with optional object:

```
var data = { x:0, y:0 };
Seatbelt1D.ForEach(m, test, data);
function test(s, extra)
{
// s is Seatbelt1D object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Seatbelt1D objects for all of the seatbelts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get seatbelts from

Return type

Array of Seatbelt1D objects

Example

To make an array of Seatbelt1D objects for all of the seatbelts in model m

```
var s = Seatbelt1D.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a seatbelt.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the seatbelt s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Seatbelt1D objects for all of the flagged seatbelts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get seatbelts from
flag	Flag	Flag set on the seatbelts that you want to retrieve

Return type

Array of Seatbelt1D objects

Example

To make an array of Seatbelt1D objects for all of the seatbelts in model m flagged with f

```
var s = Seatbelt1D.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Seatbelt1D object for a seatbelt ID.

Arguments

Name	Type	Description
Model	Model	Model to find the seatbelt in
number	integer	number of the seatbelt you want the Seatbelt1D object for

Return type

Seatbelt1D object (or null if seatbelt does not exist).

Example

To get the Seatbelt1D object for seatbelt 100 in model m

```
var s = Seatbelt1D.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Seatbelt1D property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Seatbelt1D.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	seatbelt property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Seatbelt1D property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Seatbelt1D property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this seatbelt (*ELEMENT_SEATBELT) **Note that a carriage return is not added**. See also Seatbelt1D.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for seatbelt s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the seatbelt. **Note that a carriage return is not added**. See also Seatbelt1D.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for seatbelt s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last seatbelt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last seatbelt in

Return type

Seatbelt1D object (or null if there are no seatbelts in the model).

Example

To get the last seatbelt in model m:

```
var s = Seatbelt1D.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free seatbelt label in the model. Also see Seatbelt1D.FirstFreeLabel(), Seatbelt1D.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Seatbelt1D label.

Example

To get the last free seatbelt label in model m:

```
var label = Seatbelt1D.LastFreeLabel(m);
```

Next()

Description

Returns the next seatbelt in the model.

Arguments

No arguments

Return type

Seatbelt1D object (or null if there are no more seatbelts in the model).

Example

To get the seatbelt in model m after seatbelt s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) seatbelt label in the model. Also see Seatbelt1D.FirstFreeLabel(), Seatbelt1D.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Seatbelt1D label.

Example

To get the next free seatbelt label in model m:

```
var label = Seatbelt1D.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a seatbelt.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only seatbelts from that model can be picked. If the argument is a Flag then only seatbelts that are flagged with *limit* can be selected. If omitted, or null, any seatbelts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Seatbelt1D object (or null if not picked)

Example

To pick a seatbelt from model m giving the prompt 'Pick seatbelt from screen':

```
var s = Seatbelt1D.Pick('Pick seatbelt from screen', m);
```

Previous()

Description

Returns the previous seatbelt in the model.

Arguments

No arguments

Return type

Seatbelt1D object (or null if there are no more seatbelts in the model).

Example

To get the seatbelt in model m before seatbelt s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the seatbelts in model m, from 1000000:

```
Seatbelt1D.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged seatbelts will be renumbered in
flag	Flag	Flag set on the seatbelts that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the seatbelts in model m flagged with f, from 1000000:

```
Seatbelt1D.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select seatbelts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting seatbelts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only seatbelts from that model can be selected. If the argument is a Flag then only seatbelts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any seatbelts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of seatbelts selected or null if menu cancelled

Example

To select seatbelts from model m, flagging those selected with flag f, giving the prompt ’Select seatbelts’:

```
Seatbelt1D.Select(f, ’Select seatbelts’, m);
```

To select seatbelts, flagging those selected with flag f but limiting selection to seatbelts flagged with flag l, giving the prompt ’Select seatbelts’:

```
Seatbelt1D.Select(f, ’Select seatbelts’, l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the seatbelt.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the seatbelt

Return type

No return value

Example

To set flag f for seatbelt s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the seatbelt. The seatbelt will be sketched until you either call Seatbelt1D.Unsketch(), Seatbelt1D.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the seatbelt is sketched. If omitted redraw is true. If you want to sketch several seatbelts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch seatbelt s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged seatbelts in the model. The seatbelts will be sketched until you either call Seatbelt1D.Unsketch(), Seatbelt1D.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged seatbelts will be sketched in
flag	Flag	Flag set on the seatbelts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are sketched. If omitted redraw is true. If you want to sketch flagged seatbelts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all seatbelts flagged with flag in model m:

```
Seatbelt1D.SketchFlagged(m, flag);
```

Timestep()

Description

Calculates the timestep for the seatbelt

Arguments

No arguments

Return type

real

Example

To calculate the timestep for seatbelt s:

```
var timestep = s.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing seatbelts should be counted. If false or omitted referenced but undefined seatbelts will also be included in the total.

Return type

number of seatbelts

Example

To get the total number of seatbelts in model m:

```
var total = Seatbelt1D.Total(m);
```

Unblank()

Description

Unblanks the seatbelt

Arguments

No arguments

Return type

No return value

Example

To unblank seatbelt s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the seatbelts in model m:

```
Seatbelt1D.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged seatbelts will be unblanked in
flag	Flag	Flag set on the seatbelts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the seatbelts in model m flagged with f:

```
Seatbelt1D.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all seatbelts will be unset in
flag	Flag	Flag to unset on the seatbelts

Return type

No return value

Example

To unset the flag f on all the seatbelts in model m:

```
Seatbelt1D.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the seatbelt.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the seatbelt is unsketched. If omitted redraw is true. If you want to unsketch several seatbelts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch seatbelt s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all seatbelts.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all seatbelts in model m:

```
Seatbelt1D.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unsketched in
flag	Flag	Flag set on the seatbelts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all seatbelts flagged with flag in model m:

```
Seatbelt1D.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Seatbelt1D object.

Example

To check if Seatbelt1D property s.example is a parameter by using the Seatbelt1D.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for seatbelt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for seatbelt s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this seatbelt.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for seatbelt s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the seatbelt data in keyword format. Note that this contains the keyword header and the keyword cards. See also Seatbelt1D.Keyword() and Seatbelt1D.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for seatbelt s in keyword format

```
var str = s.toString();
```

Seatbelt2D class

The Seatbelt2D class gives you access to 4 noded (2D) element seatbelt cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Seatbelt2D properties

Name	Type	Description
colour	Colour	The colour of the seatbelt
eid	integer	Seatbelt2D number. Also see the label property which is an alternative name for this.
exists	logical	true if seatbelt exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the seatbelt is in.
label	integer	Seatbelt2D number. Also see the eid property which is an alternative name for this.
model	integer	The Model number that the seatbelt is in.
n1	integer	Node 1 ID
n2	integer	Node 2 ID
n3	integer	Node 3 ID
n4	integer	Node 4 ID
pid	integer	Part ID
sbrid	integer	Retractor ID
slen	real	Initial slack length
transparency	integer	The transparency of the seatbelt (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Seatbelt2D class allows you to create, modify, edit and manipulate 4 noded element seatbelt cards. See the documentation below for more details.

Constructor

new Seatbelt2D(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*)

Description

Create a new Seatbelt2D object.

Arguments

Name	Type	Description
Model	Model	Model that seatbelt will be created in
eid	integer	Seatbelt ID.
pid	integer	Part number.
n1	integer	Node 1 ID
n2	integer	Node 2 ID
n3	integer	Node 3 ID
n4	integer	Node 4 ID

Return type

Seatbelt2D object

Example

To create a new 4 noded element seatbelt in model m with label 100, part 10 and nodes 20, 21, 22, 23:

```
var a = new Seatbelt2D(m, 100, 10, 20, 21, 22, 23);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a seatbelt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the seatbelt

Return type

No return value

Example

To associate comment c to the seatbelt s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the seatbelt

Arguments

No arguments

Return type

No return value

Example

To blank seatbelt s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the seatbelts in model m:

```
Seatbelt2D.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged seatbelts will be blanked in
flag	Flag	Flag set on the seatbelts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the seatbelts in model m flagged with f:

```
Seatbelt2D.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the seatbelt is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if seatbelt s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse seatbelt s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the seatbelt.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the seatbelt

Return type

No return value

Example

To clear flag f for seatbelt s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the seatbelt.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Seatbelt2D object

Example

To copy seatbelt s into seatbelt z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a 2 noded seatbelt.

Arguments

Name	Type	Description
Model	Model	Model that the seatbelt will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Seatbelt2D object (or null if not made)

Example

To start creating a seatbelt in model m:

```
var s = Seatbelt2D.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a seatbelt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the seatbelt

Return type

No return value

Example

To detach comment c from the seatbelt s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit seatbelt s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for seatbelt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for seatbelt s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for seatbelt.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the seatbelt colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the seatbelt.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing seatbelt s:

```
var colour = s.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first seatbelt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first seatbelt in

Return type

Seatbelt2D object (or null if there are no seatbelts in the model).

Example

To get the first seatbelt in model m:

```
var s = Seatbelt2D.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free seatbelt label in the model. Also see Seatbelt2D.LastFreeLabel(), Seatbelt2D.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Seatbelt2D label.

Example

To get the first free seatbelt label in model m:

```
var label = Seatbelt2D.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the seatbelts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be flagged in
flag	Flag	Flag to set on the seatbelts

Return type

No return value

Example

To flag all of the seatbelts with flag f in model m:

```
Seatbelt2D.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the seatbelt is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the seatbelt

Return type

true if flagged, false if not.

Example

To check if seatbelt s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each seatbelt in the model.
Note that ForEach has been designed to make looping over seatbelts as fast as possible and so has some limitations.
Firstly, a single temporary Seatbelt2D object is created and on each function call it is updated with the current seatbelt data. This means that you should not try to store the Seatbelt2D object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new seatbelts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts are in
func	function	Function to call for each seatbelt
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the seatbelts in model m:

```
Seatbelt2D.ForEach(m, test);
function test(s)
{
// s is Seatbelt2D object
}
```

To call function test for all of the seatbelts in model m with optional object:

```
var data = { x:0, y:0 };
Seatbelt2D.ForEach(m, test, data);
function test(s, extra)
{
// s is Seatbelt2D object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Seatbelt2D objects for all of the seatbelts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get seatbelts from

Return type

Array of Seatbelt2D objects

Example

To make an array of Seatbelt2D objects for all of the seatbelts in model m

```
var s = Seatbelt2D.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a seatbelt.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the seatbelt s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Seatbelt2D objects for all of the flagged seatbelts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get seatbelts from
flag	Flag	Flag set on the seatbelts that you want to retrieve

Return type

Array of Seatbelt2D objects

Example

To make an array of Seatbelt2D objects for all of the seatbelts in model m flagged with f

```
var s = Seatbelt2D.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Seatbelt2D object for a seatbelt ID.

Arguments

Name	Type	Description
Model	Model	Model to find the seatbelt in
number	integer	number of the seatbelt you want the Seatbelt2D object for

Return type

Seatbelt2D object (or null if seatbelt does not exist).

Example

To get the Seatbelt2D object for seatbelt 100 in model m

```
var s = Seatbelt2D.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Seatbelt2D property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Seatbelt2D.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	seatbelt property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Seatbelt2D property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Seatbelt2D property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this seatbelt (*ELEMENT_SEATBELT) **Note that a carriage return is not added**. See also Seatbelt2D.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for seatbelt s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the seatbelt. **Note that a carriage return is not added**. See also Seatbelt2D.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for seatbelt s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last seatbelt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last seatbelt in

Return type

Seatbelt2D object (or null if there are no seatbelts in the model).

Example

To get the last seatbelt in model m:

```
var s = Seatbelt2D.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free seatbelt label in the model. Also see Seatbelt2D.FirstFreeLabel(), Seatbelt2D.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Seatbelt2D label.

Example

To get the last free seatbelt label in model m:

```
var label = Seatbelt2D.LastFreeLabel(m);
```

Next()

Description

Returns the next seatbelt in the model.

Arguments

No arguments

Return type

Seatbelt2D object (or null if there are no more seatbelts in the model).

Example

To get the seatbelt in model m after seatbelt s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) seatbelt label in the model. Also see Seatbelt2D.FirstFreeLabel(), Seatbelt2D.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free seatbelt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Seatbelt2D label.

Example

To get the next free seatbelt label in model m:

```
var label = Seatbelt2D.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a seatbelt.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only seatbelts from that model can be picked. If the argument is a Flag then only seatbelts that are flagged with *limit* can be selected. If omitted, or null, any seatbelts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Seatbelt2D object (or null if not picked)

Example

To pick a seatbelt from model m giving the prompt 'Pick seatbelt from screen':

```
var s = Seatbelt2D.Pick('Pick seatbelt from screen', m);
```

Previous()

Description

Returns the previous seatbelt in the model.

Arguments

No arguments

Return type

Seatbelt2D object (or null if there are no more seatbelts in the model).

Example

To get the seatbelt in model m before seatbelt s:

```
var s = s.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select seatbelts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting seatbelts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only seatbelts from that model can be selected. If the argument is a Flag then only seatbelts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any seatbelts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of seatbelts selected or null if menu cancelled

Example

To select seatbelts from model m, flagging those selected with flag f, giving the prompt 'Select seatbelts':

```
Seatbelt2D.Select(f, 'Select seatbelts', m);
```

To select seatbelts, flagging those selected with flag f but limiting selection to seatbelts flagged with flag l, giving the prompt 'Select seatbelts':

```
Seatbelt2D.Select(f, 'Select seatbelts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the seatbelt.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the seatbelt

Return type

No return value

Example

To set flag f for seatbelt s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the seatbelt. The seatbelt will be sketched until you either call Seatbelt2D.Unsketch(), Seatbelt2D.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the seatbelt is sketched. If omitted redraw is true. If you want to sketch several seatbelts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch seatbelt s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged seatbelts in the model. The seatbelts will be sketched until you either call Seatbelt2D.Unsketch(), Seatbelt2D.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged seatbelts will be sketched in
flag	Flag	Flag set on the seatbelts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are sketched. If omitted redraw is true. If you want to sketch flagged seatbelts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all seatbelts flagged with flag in model m:

```
Seatbelt2D.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing seatbelts should be counted. If false or omitted referenced but undefined seatbelts will also be included in the total.

Return type

number of seatbelts

Example

To get the total number of seatbelts in model m:

```
var total = Seatbelt2D.Total(m);
```

Unblank()

Description

Unblanks the seatbelt

Arguments

No arguments

Return type

No return value

Example

To unblank seatbelt s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the seatbelts in model m:

```
Seatbelt2D.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged seatbelts will be unblanked in
flag	Flag	Flag set on the seatbelts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the seatbelts in model m flagged with f:

```
Seatbelt2D.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all seatbelts will be unset in
flag	Flag	Flag to unset on the seatbelts

Return type

No return value

Example

To unset the flag f on all the seatbelts in model m:

```
Seatbelt2D.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the seatbelt.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the seatbelt is unsketched. If omitted redraw is true. If you want to unsketch several seatbelts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch seatbelt s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all seatbelts.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all seatbelts in model m:

```
Seatbelt2D.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged seatbelts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all seatbelts will be unsketched in
flag	Flag	Flag set on the seatbelts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the seatbelts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all seatbelts flagged with flag in model m:

```
Seatbelt2D.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Seatbelt2D object.

Example

To check if Seatbelt2D property s.example is a parameter by using the Seatbelt2D.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for seatbelt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for seatbelt s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this seatbelt.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for seatbelt s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the seatbelt data in keyword format. Note that this contains the keyword header and the keyword cards. See also Seatbelt2D.Keyword() and Seatbelt2D.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for seatbelt s in keyword format

```
var str = s.toString();
```

Sensor class

The Sensor class gives you access to seatbelt sensor cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Sensor properties

Name	Type	Description
acc	real	Activating acceleration.
atime	real	Time over which acceleration must be exceeded.
colour	Colour	The colour of the sensor
dmn	real	Minimum distance
dmx	real	Maximum distance
dof	integer	Degree of freedom.
exists	logical	true if sensor exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the sensor is in.
label	integer	Sensor number. Also see the sbacid property which is an alternative name for this.
model	integer	The Model number that the sensor is in.
nid	integer	Node number.
nid1	integer	Node number 1
nid2	integer	Node number 2
pulmn	real	Maximum pull-out
pulmx	real	Maximum pull-out
pulrat	real	Rate of pull-out (length/time units)
pultim	real	Time over which rate of pull#out must be exceeded
sbrid	integer	Retractor number (for sbstyp = 2 OR 5).
sbsfl	integer	Sensor flag.
sbsid	integer	Sensor number. Also see the label property which is an alternative name for this.
sbstyp	integer	Sensor type.
time	real	Time at which sensor triggers
transparency	integer	The transparency of the sensor (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Sensor class allows you to create, modify, edit and manipulate seatbelt sensor cards. See the documentation below for more details.

Constructor

new Sensor(Model*[Model]*, sbsid*[integer]*, sbstyp*[integer]*, sbsfl (optional)*[integer]*, nid (optional)*[integer]*, nid2 (optional)*[integer]*)

Description

Create a new Seatbelt Sensor object.

Arguments

Name	Type	Description
Model	Model	Model that sensor will be created in

sbsid	integer	Sensor number.
sbstyp	integer	Sensor type
sbsfl (optional)	integer	Sensor flag. Default 0.
nid (optional)	integer	Optional node ID: Compulsory for types 1 and 4.
nid2 (optional)	integer	Optional node ID 2: Compulsory for type 4.

Return type

Sensor object

Example

To create a new seatbelt sensor in model m with label 100, type 1 and node 1:

```
var s = new Sensor(m, 100, 1, 0, 1);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a sensor.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the sensor

Return type

No return value

Example

To associate comment c to the sensor s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the sensor

Arguments

No arguments

Return type

No return value

Example

To blank sensor s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sensors in model m:

```
Sensor.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sensors will be blanked in
flag	Flag	Flag set on the sensors that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sensors in model m flagged with f:

```
Sensor.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the sensor is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if sensor s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse sensor s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the sensor.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the sensor

Return type

No return value

Example

To clear flag f for sensor s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the sensor.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Sensor object

Example

To copy sensor s into sensor z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a sensor.

Arguments

Name	Type	Description
Model	Model	Model that the sensor will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Sensor object (or null if not made)

Example

To start creating an sensor in model m:

```
var s = Sensor.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a sensor.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the sensor

Return type

No return value

Example

To detach comment c from the sensor s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit sensor s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for sensor. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for sensor s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for sensor.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the sensor colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the sensor.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing sensor s:

```
var colour = s.ExtractColour();
```

First(Model[*Model*]) [static]

Description

Returns the first sensor in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first sensor in

Return type

Sensor object (or null if there are no sensors in the model).

Example

To get the first sensor in model m:

```
var s = Sensor.First(m);
```

FirstFreeLabel(Model[*Model*], layer (optional)[*Include number*]) [static]

Description

Returns the first free sensor label in the model. Also see Sensor.LastFreeLabel(), Sensor.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free sensor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Sensor label.

Example

To get the first free sensor label in model m:

```
var label = Sensor.FirstFreeLabel(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the sensors in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be flagged in
flag	Flag	Flag to set on the sensors

Return type

No return value

Example

To flag all of the sensors with flag f in model m:

```
Sensor.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the sensor is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the sensor

Return type

true if flagged, false if not.

Example

To check if sensor s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each sensor in the model.
Note that ForEach has been designed to make looping over sensors as fast as possible and so has some limitations.
Firstly, a single temporary Sensor object is created and on each function call it is updated with the current sensor data. This means that you should not try to store the Sensor object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new sensors inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all sensors are in
func	function	Function to call for each sensor
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the sensors in model m:

```
Sensor.ForEach(m, test);
function test(s)
{
// s is Sensor object
}
```

To call function test for all of the sensors in model m with optional object:

```
var data = { x:0, y:0 };
Sensor.ForEach(m, test, data);
function test(s, extra)
{
// s is Sensor object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Sensor objects for all of the sensors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sensors from

Return type

Array of Sensor objects

Example

To make an array of Sensor objects for all of the sensors in model m

```
var s = Sensor.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a sensor.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the sensor s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Sensor objects for all of the flagged sensors in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sensors from
flag	Flag	Flag set on the sensors that you want to retrieve

Return type

Array of Sensor objects

Example

To make an array of Sensor objects for all of the sensors in model m flagged with f

```
var s = Sensor.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Sensor object for a sensor ID.

Arguments

Name	Type	Description
Model	Model	Model to find the sensor in
number	integer	number of the sensor you want the Sensor object for

Return type

Sensor object (or null if sensor does not exist).

Example

To get the Sensor object for sensor 100 in model m

```
var s = Sensor.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Sensor property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Sensor.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	sensor property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Sensor property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Sensor property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this sensor (*ELEMENT_SEATBELT_SENSEROMETER) **Note that a carriage return is not added**. See also Sensor.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for sensor s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the sensor. **Note that a carriage return is not added**. See also Sensor.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for sensor s:

```
var cards = s.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last sensor in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last sensor in

Return type

Sensor object (or null if there are no sensors in the model).

Example

To get the last sensor in model m:

```
var s = Sensor.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free sensor label in the model. Also see Sensor.FirstFreeLabel(), Sensor.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free sensor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Sensor label.

Example

To get the last free sensor label in model m:

```
var label = Sensor.LastFreeLabel(m);
```

Next()

Description

Returns the next sensor in the model.

Arguments

No arguments

Return type

Sensor object (or null if there are no more sensors in the model).

Example

To get the sensor in model m after sensor s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) sensor label in the model. Also see Sensor.FirstFreeLabel(), Sensor.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free sensor label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Sensor label.

Example

To get the next free sensor label in model m:

```
var label = Sensor.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a sensor.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sensors from that model can be picked. If the argument is a Flag then only sensors that are flagged with *limit* can be selected. If omitted, or null, any sensors from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Sensor object (or null if not picked)

Example

To pick a sensor from model m giving the prompt 'Pick sensor from screen':

```
var s = Sensor.Pick('Pick sensor from screen', m);
```

Previous()

Description

Returns the previous sensor in the model.

Arguments

No arguments

Return type

Sensor object (or null if there are no more sensors in the model).

Example

To get the sensor in model m before sensor s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sensors in model m, from 1000000:

```
Sensor.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sensors will be renumbered in
flag	Flag	Flag set on the sensors that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sensors in model m flagged with f, from 1000000:

```
Sensor.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select sensors using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting sensors
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sensors from that model can be selected. If the argument is a Flag then only sensors that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any sensors can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of sensors selected or null if menu cancelled

Example

To select sensors from model m, flagging those selected with flag f, giving the prompt 'Select sensors':

```
Sensor.Select(f, 'Select sensors', m);
```

To select sensors, flagging those selected with flag f but limiting selection to sensors flagged with flag l, giving the prompt 'Select sensors':

```
Sensor.Select(f, 'Select sensors', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the sensor.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the sensor

Return type

No return value

Example

To set flag f for sensor s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the sensor. The sensor will be sketched until you either call Sensor.Unsketch(), Sensor.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the sensor is sketched. If omitted redraw is true. If you want to sketch several sensors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch sensor s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged sensors in the model. The sensors will be sketched until you either call Sensor.Unsketch(), Sensor.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sensors will be sketched in
flag	Flag	Flag set on the sensors that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the sensors are sketched. If omitted redraw is true. If you want to sketch flagged sensors several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all sensors flagged with flag in model m:

```
Sensor.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing sensors should be counted. If false or omitted referenced but undefined sensors will also be included in the total.

Return type

number of sensors

Example

To get the total number of sensors in model m:

```
var total = Sensor.Total(m);
```

Unblank()

Description

Unblanks the sensor

Arguments

No arguments

Return type

No return value

Example

To unblank sensor s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sensors in model m:

```
Sensor.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged sensors will be unblanked in
flag	Flag	Flag set on the sensors that you want to unblank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sensors in model m flagged with f:

```
Sensor.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all sensors will be unset in
flag	Flag	Flag to unset on the sensors

Return type

No return value

Example

To unset the flag f on all the sensors in model m:

```
Sensor.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the sensor.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the sensor is unsketched. If omitted redraw is true. If you want to unsketch several sensors and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch sensor s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all sensors.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the sensors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sensors in model m:

```
Sensor.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged sensors in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sensors will be unsketched in
flag	Flag	Flag set on the sensors that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the sensors are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sensors flagged with flag in model m:

```
Sensor.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Sensor object.

Example

To check if Sensor property s.example is a parameter by using the Sensor.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for sensor. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for sensor s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this sensor.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for sensor s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the sensor data in keyword format. Note that this contains the keyword header and the keyword cards. See also Sensor.Keyword() and Sensor.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for sensor s in keyword format

```
var str = s.toString();
```

Shell class

The Shell class gives you access to shell cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- FillHolesOnFlagged(Model*[Model]*, Flag*[Flag]*, RemeshHole*[boolean]*, pid (optional)*[integer]*, Max Hole Size (optional)*[real]*, Mesh Element size (optional)*[real]*, planarSurface (optional)*[boolean]*)
- FindShellEnd() [deprecated]
- FindShellInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, sflag (optional)*[integer]*, sthick (optional)*[integer]*)
- FindShellInit(Model*[Model]*, flag (optional)*[Flag]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- MakeConsistentNormalsFlagged(Model*[Model]*, Flag*[Flag]*, Shell label (optional)*[integer]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- PickIsoparametric(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- ReverseNormalsFlagged(Model*[Model]*, Flag*[Flag]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- Angles()
- Area()
- AspectRatio()
- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- CoordsToIsoparametric(x*[real]*, y*[real]*, z*[real]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- ElemCut(Database cross section label*[integer]*)
- Error(message*[string]*, details (optional)*[string]*)

- ExtractColour()
- FillAttachedHole(pid*[integer]*, size*[real]*)
- Flagged(flag*[Flag]*)
- GetAttachedShells(tolerance (optional)*[real]*, recursive (optional)*[boolean]*)
- GetComments()
- GetCompositeData(ipt*[integer]*)
- GetNodeIDs()
- GetNodes()
- GetParameter(prop*[string]*)
- GetShellReferenceGeometry()
- IsoparametricToCoords(s*[real]*, t*[real]*)
- Jacobian()
- Keyword()
- KeywordCards()
- Length()
- Next()
- NormalVector()
- Previous()
- RemoveCompositeData(ipt*[integer]*)
- ReverseNormal(redraw (optional)*[boolean]*)
- SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*, plyid (optional)*[integer]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Skew()
- Taper()
- TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Warpage()
- WeightingFactors(s*[real]*, t*[real]*)
- Xrefs()
- toString()

Shell constants

Name	Description
Shell.EDGE_1	Edge 1 of shell
Shell.EDGE_2	Edge 2 of shell
Shell.EDGE_3	Edge 3 of shell
Shell.EDGE_4	Edge 4 of shell

Shell properties

Name	Type	Description
beta	real	Orthotropic material base offset angle. null if the _BETA option is not set. If not null then this is the angle in degrees and the _BETA option is set. This is required to distinguish between the cases of _BETA not being used (beta === null) and _BETA being set but the angle being zero (beta === 0). Prior to version 18 _BETA was only set if beta was non-zero. This was fixed in version 18 and the test changed to beta not being null. **Note: If this option is set then mcid should be 0**
colour	Colour	The colour of the shell
composite	logical	If COMPOSITE option is set. Can be true or false
composite_long	logical	If COMPOSITE_LONG option is set. Can be true or false
dof	logical	If DOF option is set. Can be true or false

edges	constant	Bitwise code of Shell.EDGE_1, Shell.EDGE_2, Shell.EDGE_3 and Shell.EDGE_4 representing which edges of the shell are free edges (read only)
eid	integer	Shell number. Also see the label property which is an alternative name for this.
exists	logical	true if shell exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the shell is in.
label	integer	Shell number. Also see the eid property which is an alternative name for this.
mcid	integer	Material coordinate system ID. If non zero then the _MCID option is assumed. **Note: If this option is set then beta should be null.**
model	integer	The Model number that the shell is in.
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
n4	integer	Node number 4
n5	integer	Node number 5
n6	integer	Node number 6
n7	integer	Node number 7
n8	integer	Node number 8
nip	logical	Number of integration points for composite shell
nodes	integer	Number of nodes shell has (read only)
ns1	integer	Scalar Node number 1
ns2	integer	Scalar Node number 2
ns3	integer	Scalar Node number 3
ns4	integer	Scalar Node number 4
offset	real	Offset distance. If non zero then the _OFFSET option is assumed
pid	integer	Part number
shl4_to_shl8	logical	If SHL4_TO_SHL8 option is set. Can be true or false
thic1	real	Thickness at node 1
thic2	real	Thickness at node 2
thic3	real	Thickness at node 3
thic4	real	Thickness at node 4
thic5	real	Thickness at node 5 (if 8 noded shell)
thic6	real	Thickness at node 6 (if 8 noded shell)
thic7	real	Thickness at node 7 (if 8 noded shell)
thic8	real	Thickness at node 8 (if 8 noded shell)
thickness	logical	If _THICKNESS option is set. Can be true or false
transparency	integer	The transparency of the shell (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Shell class allows you to create, modify, edit and manipulate shell cards. See the documentation below for more details.

Constructor

new Shell(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4 (optional)*[integer]*, n5 (optional)*[integer]*, n6 (optional)*[integer]*, n7 (optional)*[integer]*, n8 (optional)*[integer]*)

Description

Create a new Shell object. Use either 3, 4, 6 or 8 nodes when creating a new shell. If you are creating a 3 noded shell either only give 3 nodes or give 4 nodes but make nodes 3 and 4 the same number. Similarly, 6 noded shells can be created with 6 node arguments or with 8 nodes but nodes 3 and 4 the same number and nodes 7 and 8 the same number.

Arguments

Name	Type	Description
Model	Model	Model that shell will be created in
eid	integer	Shell number
pid	integer	Part number
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
n4 (optional)	integer	Node number 4
n5 (optional)	integer	Node number 5
n6 (optional)	integer	Node number 6
n7 (optional)	integer	Node number 7
n8 (optional)	integer	Node number 8

Return type

Shell object

Example

To create a new shell in model m with label 100, part 10 and nodes 1, 2, 3, 4:

```
var s = new Shell(m, 100, 10, 1, 2, 3, 4);
```

Details of functions

Angles()

Description

Calculates the minimum and maximum internal angles (in degrees) for the shell

Arguments

No arguments

Return type

Array of numbers containing min and max angles

Example

To calculate the maximum and minimum internal angles for shell s:

```
var angles = s.Angles();
var min = angles[0];
var max = angles[1];
```

Area()

Description

Calculates the area for the shell

Arguments

No arguments

Return type

real

Example

To calculate the area for shell s:

```
var area = s.Area();
```

AspectRatio()

Description

Calculates the aspect ratio for the shell

Arguments

No arguments

Return type

real

Example

To calculate the aspect ratio for shell s:

```
var ratio = s.AspectRatio();
```

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the shell

Return type

No return value

Example

To associate comment c to the shell s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the shell

Arguments

No arguments

Return type

No return value

Example

To blank shell s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the shells in model m:

```
Shell.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged shells will be blanked in
flag	Flag	Flag set on the shells that you want to blank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the shells in model m flagged with f:

```
Shell.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the shell is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if shell s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse shell s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the shell.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the shell

Return type

No return value

Example

To clear flag f for shell s:

```
s.ClearFlag(f);
```

CoordsToIsoparametric(x*[real]*, y*[real]*, z*[real]*)

Description

Calculates the isoparametric coordinates for a point on the shell.

Arguments

Name	Type	Description
x	real	X coordinate of point
y	real	Y coordinate of point
z	real	Z coordinate of point

Return type

Array containing s and t isoparametric coordinates and the distance the point is from the shell (positive in direction of shell normal). If it is not possible to calculate the isoparametric coordinates null is returned.

Example

To calculate the isoparametric coordinates of point (10, 20, 30) on shell s:

```
var isocoords = s.CoordsToIsoparametric(10, 20, 30);
```

Copy(range (optional)*[boolean]*)

Description

Copies the shell.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Shell object

Example

To copy shell s into shell z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a shell.

Arguments

Name	Type	Description
Model	Model	Model that the shell will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Shell object (or null if not made)

Example

To start creating a shell in model m:

```
var s = Shell.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the shell

Return type

No return value

Example

To detach comment c from the shell s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit shell s:

```
s.Edit();
```

ElemCut(Database cross section label*[integer]*)

Description

Returns coordinates of the intersections between a shell and a database cross section.

Arguments

Name	Type	Description
Database cross section label	integer	The label of the database cross section.

Return type

An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line, or NULL if it does not cut. Note this function does not check that the shell is in the cross section definition (part set)

Example

To get the cut line coordinates between database cross section 200 and shell s:

```
var data = s.ElemCut(200)
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for shell s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for shell.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the shell colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the shell.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing shell s:

```
var colour = s.ExtractColour();
```

FillAttachedHole(pid*[integer]*, size*[real]*)

Description

Fills in (meshes) a hole attached to the shell.

Arguments

Name	Type	Description
pid	integer	The Part number that the new shells will be created in.
size	real	The size for created elements.

Return type

No return value.

Example

To fill in a hole attached to shell s, putting new shells with size 5.0 into part 100:

```
s.FillAttachedHole(100, 5.0);
```

FillHolesOnFlagged(Model*[Model]*, Flag*[Flag]*, RemeshHole*[boolean]*, pid (optional)*[integer]*, Max Hole Size (optional)*[real]*, Mesh Element size (optional)*[real]*, planarSurface (optional)*[boolean]*) [static]

Description

Fills multiple holes using flagged shells.

Arguments

Name	Type	Description
Model	Model	Model that all shells are in.
Flag	Flag	flag bit
RemeshHole	boolean	TRUE if elements around the hole should be remeshed
pid (optional)	integer	Needs to be specified if RemeshHole is FALSE. Specifies the Part id where the mesh is filled
Max Hole Size (optional)	real	Maximum size of the hole which is to be filled. If omitted a default size of 20.0 will be set
Mesh Element size (optional)	real	Element size of the mesh which fills the hole. If omitted a default size of 10.0 will be set
planarSurface (optional)	boolean	Needs to be specified if RemeshHole is TRUE. TRUE if we need to Use planar surface

Return type

No return value.

Example

To fill holes on flagged shells:

```
Shell.FillHolesOnFlagged(m, flag, 1, 112, 60.5, 5.34 ,0);
```

Note: pid is required when RemeshHole is FALSE

```
Shell.FillHolesOnFlagged(m, flag, 0, 112);
```

FindShellEnd() [static] [deprecated]

This function is deprecated in version 16.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Tidy memory allocation incurred by function which finds shells within a box. Now replaced by model member function Model.FindElemEnd()

Arguments

No arguments

Return type

No return value

Example

```
Shell.FindShellEnd();
```

FindShellInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, sflag (optional)*[integer]*, sthick (optional)*[integer]*) [static]

Description

Returns an array of Shell objects for the shells within a box. This requires a previous (outside loop) call to function FindShellInit(m) or m.FindElemInit() where the process is initialized for flagged shells in the model (typically all shells) and m.FindElemEnd() to close the process. Please note this function provides a list of all shells that could potentially be in the box (using computationally cheap bounding box comparison) it is not a rigorous test of whether the shell is actually in the box. See also Shell.FindShellInit() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model designated model
xmin	real	Minimum bound in global x
xmax	real	Maximum bound in global x
ymin	real	Minimum bound in global y
ymax	real	Maximum bound in global y
zmin	real	Minimum bound in global z
zmax	real	Maximum bound in global z
sflag (optional)	integer	Optional flag to restrict shells considered
sthick (optional)	integer	Optional flag to consider thickness for shells

Return type

Array of Shell objects

Example

To get an array of Shell objects for shells in model m within defined box.

```
Shell.FindShellInit(m);
```

or clear model flag and flag elements of interest and

```
m.FindElemInit(flag);

{

    //loop in which boxes are formed and tested

    //find shells both in box and flagged with sflag

    //consider shell thickness

    var s = Shell.FindShellInBox(m, xmin, xmax, ymin, ymax, zmin, zmax, sflag,
1);

    if(s.length) ...

}

m.FindElemEnd();
```

FindShellInit(Model*[Model]*, flag (optional)*[Flag]*) [static]

Description

Initialize setup so that all flagged shells in model can be tested to see if they are within box. See also Shell.FindShellInBox() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model in which shells have been flagged
flag (optional)	Flag	Optional flag that has been set on the shells, if unsupplied all shells considered

Return type

No return value

Example

To initialize find setup for all shells in model m:

```
Shell.FindShellInit(m);
```

First(Model*[Model]*) [static]

Description

Returns the first shell in the model.

Arguments

Name	Type	Description

Model	Model	Model to get first shell in

Return type

Shell object (or null if there are no shells in the model).

Example

To get the first shell in model m:

```
var s = Shell.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free shell label in the model. Also see Shell.LastFreeLabel(), Shell.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Shell label.

Example

To get the first free shell label in model m:

```
var label = Shell.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the shells in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be flagged in
flag	Flag	Flag to set on the shells

Return type

No return value

Example

To flag all of the shells with flag f in model m:

```
Shell.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the shell is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the shell

Return type

true if flagged, false if not.

Example

To check if shell s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each shell in the model.
Note that ForEach has been designed to make looping over shells as fast as possible and so has some limitations. Firstly, a single temporary Shell object is created and on each function call it is updated with the current shell data. This means that you should not try to store the Shell object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new shells inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all shells are in
func	function	Function to call for each shell
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the shells in model m:

```
Shell.ForEach(m, test);
function test(s)
{
// s is Shell object
}
```

To call function test for all of the shells in model m with optional object:

```
var data = { x:0, y:0 };
Shell.ForEach(m, test, data);
function test(s, extra)
{
// s is Shell object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Shell objects for all of the shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get shells from

Return type

Array of Shell objects

Example

To make an array of Shell objects for all of the shells in model m

```
var s = Shell.GetAll(m);
```

GetAttachedShells(tolerance (optional)*[real]*, recursive (optional)*[boolean]*)

Description

Returns the shells that are attached to the shell. **Note that 'attached' means that the shells must share 2 nodes**.

Arguments

Name	Type	Description
tolerance (optional)	real	This tolerance can be used to limit the selection to shells whose normal vector is within this tolerance (in degrees) of the original shell. If omitted the tolerance is 180 degrees.
recursive (optional)	boolean	If recursive is false then only the shells actually attached to the shell will be returned (this could also be done by using the Xrefs class but this method is provided for convenience. If recursive is true then PRIMER will keep finding attached shells until no more can be found. If omitted recursive will be false.

Return type

Array of Shell objects (or null if there are no attached shells).

Example

To find the shells attached to shell s with a 10 degree tolerance, growing the selection until no more shells can be found:

```
var shell_array = s.GetAttachedShells(10, true);
```

GetComments()

Description

Extracts the comments associated to a shell.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the shell s:

```
var comm_array = s.GetComments();
```

GetCompositeData(ipt*[integer]*)

Description

Returns the composite data for an integration point in *ELEMENT_SHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want the data for. **Note that integration points start at 0, not 1**.

Return type

An array containing the material ID, thickness and beta angle values. If the _COMPOSITE_LONG option is set, then the array returned will also contain the ply ID.

Example

To get the composite data for the 3rd integration point for shell s:

```
if (s.composite && s.nip >= 3)
{
    var ipt_data = s.GetCompositeData(2);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Shell objects for all of the flagged shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get shells from
flag	Flag	Flag set on the shells that you want to retrieve

Return type

Array of Shell objects

Example

To make an array of Shell objects for all of the shells in model m flagged with f

```
var s = Shell.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Shell object for a shell ID.

Arguments

Name	Type	Description
Model	Model	Model to find the shell in
number	integer	number of the shell you want the Shell object for

Return type

Shell object (or null if shell does not exist).

Example

To get the Shell object for shell 100 in model m

```
var s = Shell.GetFromID(m, 100);
```

GetNodeIDs()

Description

Returns the labels of the nodes on the shell as an array. See also Shell.GetNodes()

Arguments

No arguments

Return type

Array of node labels (integers)

Example

To return the node labels of shell s as an array

```
var nodes = s.GetNodeIDs();
```

GetNodes()

Description

Returns the nodes on the shell as an array of Node objects. See also Shell.GetNodeIDs()

Arguments

No arguments

Return type

Array of Node objects

Example

To return the nodes of shell s as an array

```
var nodes = s.GetNodes();
```

GetParameter(prop*[string]*)

Description

Checks if a Shell property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Shell.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	shell property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Shell property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Shell property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

GetShellReferenceGeometry()

Description

Returns the airbag shell reference geometry of the shell

Arguments

No arguments

Return type

The shell reference geometry ID of the shell (or 0 if it hasn't got any)

Example

To get the shell reference geometry of the shell s:

```
var a = s.GetShellReferenceGeometry();
```

IsoparametricToCoords(s*[real]*, t*[real]*)

Description

Calculates the coordinates for a point on the shell from the isoparametric coords.

Arguments

Name	Type	Description
s	real	First isoparametric coordinate

t	real	Second isoparametric coordinate

Return type

Array of numbers containing x, y and z or null if not possible to calculate.

Example

To calculate the coordinates of isoparametric point (0.5, -0.5) on shell s:

```
var coords = s.IsoparametricToCoords(0.5, -0.5);
```

Jacobian()

Description

Calculates the jacobian for the shell

Arguments

No arguments

Return type

real

Example

To calculate the jacobian for shell s:

```
var jacobian = s.Jacobian();
```

Keyword()

Description

Returns the keyword for this shell (*SHELL, *SHELL_SCALAR or *SHELL_SCALAR_VALUE). **Note that a carriage return is not added**. See also Shell.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for shell s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the shell. **Note that a carriage return is not added**. See also Shell.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for shell s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last shell in

Return type

Shell object (or null if there are no shells in the model).

Example

To get the last shell in model m:

```
var s = Shell.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free shell label in the model. Also see Shell.FirstFreeLabel(), Shell.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Shell label.

Example

To get the last free shell label in model m:

```
var label = Shell.LastFreeLabel(m);
```

Length()

Description

Calculates the minimum length for the shell

Arguments

No arguments

Return type

real

Example

To calculate the minimum length for shell s:

```
var length = s.Length();
```

MakeConsistentNormalsFlagged(Model*[Model]*, Flag*[Flag]*, Shell label (optional)*[integer]*) [static]

Description

Make all the flagged SHELL normals consistent with a selected one, the Seed Element.

Arguments

Name	Type	Description
Model	Model	Model that all shells are in.
Flag	Flag	flag bit
Shell label (optional)	integer	The label of the seed shell. If omitted, or null, the first flagged shell is used as the seed shell.

Return type

Array containing the labels of shells which have had normals reversed

Example

To make all flagged shell normals consistent:

```
Shell.MakeConsistentNormalsFlagged(m, flag, 1001);
```

Next()

Description

Returns the next shell in the model.

Arguments

No arguments

Return type

Shell object (or null if there are no more shells in the model).

Example

To get the shell in model m after shell s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) shell label in the model. Also see Shell.FirstFreeLabel(), Shell.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Shell label.

Example

To get the next free shell label in model m:

```
var label = Shell.NextFreeLabel(m);
```

NormalVector()

Description

Calculates the unit normal vector for the shell.

Arguments

No arguments

Return type

Array of numbers containing x, y and z components of unit normal vector or null if the vector cannot be calculated (for example if the shell has zero area).

Example

To calculate the normal vector of shell s:

```
var nvector = s.NormalVector();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a shell.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only shells from that model can be picked. If the argument is a Flag then only shells that are flagged with *limit* can be selected. If omitted, or null, any shells from any model can be selected. from any model.

modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Shell object (or null if not picked)

Example

To pick a shell from model m giving the prompt 'Pick shell from screen':

```
var s = Shell.Pick('Pick shell from screen', m);
```

PickIsoparametric(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a point on a shell. The isoparametric coordinates of the point picked on the shell are returned as well as the shell picked. These coordinates are suitable for using in the function Shell.IsoparametricToCoords(). See also Shell.Pick()

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only shells from that model can be picked. If the argument is a Flag then only shells that are flagged with *limit* can be selected. If omitted, or null, any shells from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Array containing Shell object and isoparametric coordinates (or null if not picked or the point is not on a shell)

Example

To pick a point on a shell from model m giving the prompt 'Pick a point on a shell on the screen':

```
var a = Shell.PickIsoparametric('Pick a point on a shell on the screen', m);
if (a != null)
{
    Message("You picked point "+a[1]+","+a[2]+" on shell "+a[0].label);
}
```

Previous()

Description

Returns the previous shell in the model.

Arguments

No arguments

Return type

Shell object (or null if there are no more shells in the model).

Example

To get the shell in model m before shell s:

```
var s = s.Previous();
```

RemoveCompositeData(ipt*[integer]*)

Description

Removes the composite data for an integration point in *ELEMENT_SHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to remove. **Note that integration points start at 0, not 1**.

Return type

No return value.

Example

To remove the composite data for the 3rd integration point for shell s:

```
s.RemoveCompositeData(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the shells in model m, from 1000000:

```
Shell.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged shells will be renumbered in
flag	Flag	Flag set on the shells that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the shells in model m flagged with f, from 1000000:

```
Shell.RenumberFlagged(m, f, 1000000);
```

ReverseNormal(redraw (optional)*[boolean]*)

Description

Reverse shell normal.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to reverse several shell normals and only redraw after the last one then use false for all redraws apart from the last one.

Return type

No return value.

Example

To Reverse shell normal for shell s:

```
s.ReverseNormal();
```

ReverseNormalsFlagged(Model*[Model]*, Flag*[Flag]*) [static]

Description

Reverse all the flagged shell normals

Arguments

Name	Type	Description
Model	Model	Model that all shells are in.
Flag	Flag	flag bit

Return type

No return value.

Example

To Reverse all flagged shell normals:

```
Shell.ReverseNormalsFlagged(m, flag);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select shells using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting shells
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only shells from that model can be selected. If the argument is a Flag then only shells that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any shells can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of shells selected or null if menu cancelled

Example

To select shells from model m, flagging those selected with flag f, giving the prompt 'Select shells':

```
Shell.Select(f, 'Select shells', m);
```

To select shells, flagging those selected with flag f but limiting selection to shells flagged with flag l, giving the prompt 'Select shells':

```
Shell.Select(f, 'Select shells', l);
```

SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*, plyid (optional)*[integer]*)

Description

Sets the composite data for an integration point in *ELEMENT_SHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to set the data for. **Note that integration points start at 0, not 1**.
mid	integer	Material ID for the integration point.
thick	real	Thickness of the integration point.
beta	real	Material angle of the integration point.
plyid (optional)	integer	Ply ID for the integration point. This should be used if the _COMPOSITE_LONG option is set for the shell.

Return type

No return value.

Example

To set the composite data for the 3rd integration point to mat 1, thickness 0.5 and angle 45, for shell s:

```
s.SetCompositeData(2, 1, 0.5, 45);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the shell.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the shell

Return type

No return value

Example

To set flag f for shell s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the shell. The shell will be sketched until you either call Shell.Unsketch(), Shell.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the shell is sketched. If omitted redraw is true. If you want to sketch several shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch shell s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged shells in the model. The shells will be sketched until you either call Shell.Unsketch(), Shell.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description

Model	Model	Model that all the flagged shells will be sketched in
flag	Flag	Flag set on the shells that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the shells are sketched. If omitted redraw is true. If you want to sketch flagged shells several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all shells flagged with flag in model m:

```
Shell.SketchFlagged(m, flag);
```

Skew()

Description

Calculates the skew for the shell

Arguments

No arguments

Return type

real

Example

To calculate the skew for shell s:

```
var skew = s.Skew();
```

Taper()

Description

Calculates the taper for the shell

Arguments

No arguments

Return type

real

Example

To calculate the taper for shell s:

```
var taper = s.Taper();
```

TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)

Description

Checks if nodes of shell are tied by contact or directly attached (non-zero option1).

Arguments

Name	Type	Description
Contact label	integer	The label of the tied contact. If zero the tied contact is found for the shell by reverse lookup.
Flag	Flag	flag bit
Option1	integer	Directly tied node (logical OR) 0:NONE 1:NRB/C_EXNO 2:BEAM 4:SHELL 8:SOLID 16:TSHELL
Option2	integer	0:No action 1: report error if directly attached node (acc. option1) captured by contact

Return type

string

Example

To check if all nodes of shell s are tied by contact 200 or attach directly to constraint:

```
var message = s.TiedNodeCheck(200, flag, 1, 1)
```

Timestep()

Description

Calculates the timestep for the shell

Arguments

No arguments

Return type

real

Example

To calculate the timestep for shell s:

```
var timestep = s.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of shells in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing shells should be counted. If false or omitted referenced but undefined shells will also be included in the total.

Return type

number of shells

Example

To get the total number of shells in model m:

```
var total = Shell.Total(m);
```

Unblank()

Description

Unblanks the shell

Arguments

No arguments

Return type

No return value

Example

To unblank shell s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the shells in model m:

```
Shell.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged shells will be unblanked in
flag	Flag	Flag set on the shells that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the shells in model m flagged with f:

```
Shell.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all shells will be unset in
flag	Flag	Flag to unset on the shells

Return type

No return value

Example

To unset the flag f on all the shells in model m:

```
Shell.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the shell.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the shell is unsketched. If omitted redraw is true. If you want to unsketch several shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch shell s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all shells.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all shells in model m:

```
Shell.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all shells will be unsketched in
flag	Flag	Flag set on the shells that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all shells flagged with flag in model m:

```
Shell.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Shell object.

Example

To check if Shell property s.example is a parameter by using the Shell.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for shell s:

```
s.Warning("My custom warning");
```

Warpage()

Description

Calculates the warpage for the shell

Arguments

No arguments

Return type

real

Example

To calculate the warpage for shell s:

```
var warpage = s.Warpage();
```

WeightingFactors(s*[real]*, t*[real]*)

Description

Calculates the weighting factors for a point on the shell from the isoparametric coords.

Arguments

Name	Type	Description
s	real	First isoparametric coordinate
t	real	Second isoparametric coordinate

Return type

Array of numbers containing weighting factors or null if not possible to calculate.

Example

To calculate the weighting factors of isoparametric point (0.5, -0.5) on shell s:

```
var weights = s.WeightingFactors(0.5, -0.5);
```

Xrefs()

Description

Returns the cross references for this shell.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for shell s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the shell data in keyword format. Note that this contains the keyword header and the keyword cards. See also Shell.Keyword() and Shell.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for shell s in keyword format

```
var str = s.toString();
```

Slipring class

The Slipring class gives you access to seatbelt slipring cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Slipring properties

Name	Type	Description
colour	Colour	The colour of the slipring
dc	real	Optional decay constant to allow a smooth transition between the static and dynamic friction coefficients.
direct	integer	Direction of belt movement
exists	logical	true if slipring exists, false if referred to but not defined. (read only)
fc	real	Coulomb dynamic friction coefficient
fcs	real	Coulomb static friction coefficient
funcid	integer	Function ID to determine friction coefficient
include	integer	The Include file number that the slipring is in.
k	real	Optional coefficient for determining the Coulomb friction coefficient related to angle alpha
label	integer	Slipring number. Also see the sbsrid property which is an alternative name for this.
lcnffd	integer	Loadcurve for Coulomb dynamic friction
lcnffs	integer	Loadcurve for Coulomb static friction
ltime	real	Slipring lockup time
model	integer	The Model number that the slipring is in.
onid	integer	Orientation Node number
sbid1	integer	Seatbelt number 1 (or Set Shell number if sbrnid is negative).
sbid2	integer	Seatbelt number 2 (or Set Shell number if sbrnid is negative).
sbrnid	integer	Node number (or Set Node number if negative)
sbsrid	integer	Slipring number. Also see the label property which is an alternative name for this.
shell_ seatbelt	logical	true if slipring is used for shell (2D) seatbelt elements. (read only)
transparency	integer	The transparency of the slipring (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Slipring class allows you to create, modify, edit and manipulate seatbelt slipring cards. See the documentation below for more details.

Constructor

new Slipring(Model*[Model]*, sbsrid*[integer]*, sbid1*[integer]*, sbid2*[integer]*, sbrnid*[integer]*)

Description

Create a new Seatbelt Slipring object.

Arguments

Name	Type	Description
Model	Model	Model that slipring will be created in

sbsrid	integer	Slipring number.
sbid1	integer	Seatbelt number 1
sbid2	integer	Seatbelt number 2
sbrnid	integer	Slipring Node number

Return type

Slipring object

Example

To create a new seatbelt slipring in model m with label 100, seatbelts 10, 11 and node 20:

```
var a = new Slipring(m, 100, 10, 11, 20);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a slipring.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the slipring

Return type

No return value

Example

To associate comment c to the slipring s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the slipring

Arguments

No arguments

Return type

No return value

Example

To blank slipring s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sliprings in model m:

```
Slipring.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sliprings will be blanked in
flag	Flag	Flag set on the sliprings that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sliprings in model m flagged with f:

```
Slipring.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the slipring is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if slipring s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse slipring s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the slipring.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the slipring

Return type

No return value

Example

To clear flag f for slipring s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the slipring.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Slipring object

Example

To copy slipring s into slipring z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a slipring.

Arguments

Name	Type	Description
Model	Model	Model that the slipring will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Slipring object (or null if not made)

Example

To start creating an slipring in model m:

```
var s = Slipring.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a slipring.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the slipring

Return type

No return value

Example

To detach comment c from the slipring s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit slipring s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for slipring. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for slipring s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for slipring.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the slipring colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the slipring.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing slipring s:

```
var colour = s.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first slipring in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first slipring in

Return type

Slipring object (or null if there are no sliprings in the model).

Example

To get the first slipring in model m:

```
var s = Slipring.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free slipring label in the model. Also see Slipring.LastFreeLabel(), Slipring.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free slipring label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Slipring label.

Example

To get the first free slipring label in model m:

```
var label = Slipring.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the sliprings in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be flagged in
flag	Flag	Flag to set on the sliprings

Return type

No return value

Example

To flag all of the sliprings with flag f in model m:

```
Slipring.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the slipring is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the slipring

Return type

true if flagged, false if not.

Example

To check if slipring s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each slipring in the model.
Note that ForEach has been designed to make looping over sliprings as fast as possible and so has some limitations.
Firstly, a single temporary Slipring object is created and on each function call it is updated with the current slipring data. This means that you should not try to store the Slipring object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new sliprings inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings are in
func	function	Function to call for each slipring
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the sliprings in model m:

```
Slipring.ForEach(m, test);
function test(s)
{
// s is Slipring object
}
```

To call function test for all of the sliprings in model m with optional object:

```
var data = { x:0, y:0 };
Slipring.ForEach(m, test, data);
function test(s, extra)
{
// s is Slipring object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Slipring objects for all of the sliprings in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sliprings from

Return type

Array of Slipring objects

Example

To make an array of Slipring objects for all of the sliprings in model m

```
var s = Slipring.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a slipring.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the slipring s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Slipring objects for all of the flagged sliprings in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sliprings from
flag	Flag	Flag set on the sliprings that you want to retrieve

Return type

Array of Slipring objects

Example

To make an array of Slipring objects for all of the sliprings in model m flagged with f

```
var s = Slipring.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Slipring object for a slipring ID.

Arguments

Name	Type	Description
Model	Model	Model to find the slipring in
number	integer	number of the slipring you want the Slipring object for

Return type

Slipring object (or null if slipring does not exist).

Example

To get the Slipring object for slipring 100 in model m

```
var s = Slipring.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Slipring property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Slipring.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	slipring property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Slipring property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Slipring property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this slipring (*ELEMENT_SEATBELT_SLIPEROMETER) **Note that a carriage return is not added**. See also Slipring.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for slipring s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the slipring. **Note that a carriage return is not added**. See also Slipring.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for slipring s:

```
var cards = s.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last slipring in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last slipring in

Return type

Slipring object (or null if there are no sliprings in the model).

Example

To get the last slipring in model m:

```
var s = Slipring.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free slipring label in the model. Also see Slipring.FirstFreeLabel(), Slipring.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free slipring label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Slipring label.

Example

To get the last free slipring label in model m:

```
var label = Slipring.LastFreeLabel(m);
```

Next()

Description

Returns the next slipring in the model.

Arguments

No arguments

Return type

Slipring object (or null if there are no more sliprings in the model).

Example

To get the slipring in model m after slipring s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) slipring label in the model. Also see Slipring.FirstFreeLabel(), Slipring.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free slipring label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Slipring label.

Example

To get the next free slipring label in model m:

```
var label = Slipring.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a slipring.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sliprings from that model can be picked. If the argument is a Flag then only sliprings that are flagged with *limit* can be selected. If omitted, or null, any sliprings from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Slipring object (or null if not picked)

Example

To pick a slipring from model m giving the prompt 'Pick slipring from screen':

```
var s = Slipring.Pick('Pick slipring from screen', m);
```

Previous()

Description

Returns the previous slipring in the model.

Arguments

No arguments

Return type

Slipring object (or null if there are no more sliprings in the model).

Example

To get the slipring in model m before slipring s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sliprings in model m, from 1000000:

```
Slipring.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sliprings will be renumbered in
flag	Flag	Flag set on the sliprings that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sliprings in model m flagged with f, from 1000000:

```
Slipring.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select sliprings using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting sliprings
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sliprings from that model can be selected. If the argument is a Flag then only sliprings that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any sliprings can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of sliprings selected or null if menu cancelled

Example

To select sliprings from model m, flagging those selected with flag f, giving the prompt 'Select sliprings':

```
Slipring.Select(f, 'Select sliprings', m);
```

To select sliprings, flagging those selected with flag f but limiting selection to sliprings flagged with flag l, giving the prompt 'Select sliprings':

```
Slipring.Select(f, 'Select sliprings', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the slipring.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the slipring

Return type

No return value

Example

To set flag f for slipring s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the slipring. The slipring will be sketched until you either call Slipring.Unsketch(), Slipring.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the slipring is sketched. If omitted redraw is true. If you want to sketch several sliprings and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch slipring s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged sliprings in the model. The sliprings will be sketched until you either call Slipring.Unsketch(), Slipring.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sliprings will be sketched in
flag	Flag	Flag set on the sliprings that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the sliprings are sketched. If omitted redraw is true. If you want to sketch flagged sliprings several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all sliprings flagged with flag in model m:

```
Slipring.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing sliprings should be counted. If false or omitted referenced but undefined sliprings will also be included in the total.

Return type

number of sliprings

Example

To get the total number of sliprings in model m:

```
var total = Slipring.Total(m);
```

Unblank()

Description

Unblanks the slipring

Arguments

No arguments

Return type

No return value

Example

To unblank slipring s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sliprings in model m:

```
Slipring.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged sliprings will be unblanked in

flag	Flag	Flag set on the sliprings that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sliprings in model m flagged with f:

```
Slipring.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all sliprings will be unset in
flag	Flag	Flag to unset on the sliprings

Return type

No return value

Example

To unset the flag f on all the sliprings in model m:

```
Slipring.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the slipring.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the slipring is unsketched. If omitted redraw is true. If you want to unsketch several sliprings and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch slipring s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all sliprings.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the sliprings are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sliprings in model m:

```
Slipring.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged sliprings in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sliprings will be unsketched in
flag	Flag	Flag set on the sliprings that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the sliprings are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sliprings flagged with flag in model m:

```
Slipring.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Slipring object.

Example

To check if Slipring property s.example is a parameter by using the Slipring.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for slipring. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for slipring s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this slipring.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for slipring s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the slipring data in keyword format. Note that this contains the keyword header and the keyword cards. See also Slipring.Keyword() and Slipring.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for slipring s in keyword format

```
var str = s.toString();
```

Solid class

The Solid class gives you access to solid cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- CoordsToIsoparametric(Model*[Model]*, x*[real]*, y*[real]*, z*[real]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- FindSolidEnd() [deprecated]
- FindSolidInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, hflag (optional)*[integer]*)
- FindSolidInit(Model*[Model]*, flag (optional)*[Flag]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- ElemCut(Database cross section label*[integer]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()

- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- TetCollapse()
- TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Volume()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Solid constants

Name	Description
Solid.EDGE_1	Edge 1 of solid
Solid.EDGE_10	Edge 10 of solid
Solid.EDGE_11	Edge 11 of solid
Solid.EDGE_12	Edge 12 of solid
Solid.EDGE_2	Edge 2 of solid
Solid.EDGE_3	Edge 3 of solid
Solid.EDGE_4	Edge 4 of solid
Solid.EDGE_5	Edge 5 of solid
Solid.EDGE_6	Edge 6 of solid
Solid.EDGE_7	Edge 7 of solid
Solid.EDGE_8	Edge 8 of solid
Solid.EDGE_9	Edge 9 of solid
Solid.FACE_1	Face 1 of solid
Solid.FACE_2	Face 2 of solid
Solid.FACE_3	Face 3 of solid
Solid.FACE_4	Face 4 of solid
Solid.FACE_5	Face 5 of solid
Solid.FACE_6	Face 6 of solid

Solid properties

Name	Type	Description
a1	real	x component of material direction a
a2	real	y component of material direction a
a3	real	z component of material direction a
colour	Colour	The colour of the solid
d1	real	x component of material in-plane vector
d2	real	y component of material in-plane vector
d3	real	z component of material in-plane vector
dof	logical	If DOF option is set. Can be true or false

edges	constant	Bitwise code of Solid.EDGE_1, Solid.EDGE_2, Solid.EDGE_3, Solid.EDGE_4, Solid.EDGE_5, Solid.EDGE_6, Solid.EDGE_7, Solid.EDGE_8, Solid.EDGE_9, Solid.EDGE_10, Solid.EDGE_11 and Solid.EDGE_12 representing which edges of the solid are free faces (read only)
eid	integer	Solid number. Also see the label property which is an alternative name for this.
exists	logical	true if solid exists, false if referred to but not defined. (read only)
faces	constant	Bitwise code of Solid.FACE_1, Solid.FACE_2, Solid.FACE_3, Solid.FACE_4, Solid.FACE_5 and Solid.FACE_6 representing which faces of the solid are internal faces (read only). Note that this is calculated from the solids that are currently visible so blanking solids will affect this property once graphics have been updated.
h20	logical	If _H20 option is set. Can be true or false
h27	logical	If _H27 option is set. Can be true or false
h64	logical	If _H64 option is set. Can be true or false
h8toh20	logical	If _H8TOH20 option is set. Can be true or false
h8toh27	logical	If _H8TOH27 option is set. Can be true or false
h8toh64	logical	If _H8TOH64 option is set. Can be true or false
include	integer	The Include file number that the solid is in.
label	integer	Solid number. Also see the eid property which is an alternative name for this.
model	integer	The Model number that the solid is in.
n1	integer	Node number 1
n10	integer	Node number 10
n11	integer	Node number 11
n12	integer	Node number 12
n13	integer	Node number 13
n14	integer	Node number 14
n15	integer	Node number 15
n16	integer	Node number 16
n17	integer	Node number 17
n18	integer	Node number 18
n19	integer	Node number 19
n2	integer	Node number 2
n20	integer	Node number 20
n21	integer	Node number 21
n22	integer	Node number 22
n23	integer	Node number 23
n24	integer	Node number 24
n25	integer	Node number 25
n26	integer	Node number 26
n27	integer	Node number 27
n28	integer	Node number 28
n29	integer	Node number 29

n3	integer	Node number 3
n30	integer	Node number 30
n31	integer	Node number 31
n32	integer	Node number 32
n33	integer	Node number 33
n34	integer	Node number 34
n35	integer	Node number 35
n36	integer	Node number 36
n37	integer	Node number 37
n38	integer	Node number 38
n39	integer	Node number 39
n4	integer	Node number 4
n40	integer	Node number 40
n41	integer	Node number 41
n42	integer	Node number 42
n43	integer	Node number 43
n44	integer	Node number 44
n45	integer	Node number 45
n46	integer	Node number 46
n47	integer	Node number 47
n48	integer	Node number 48
n49	integer	Node number 49
n5	integer	Node number 5
n50	integer	Node number 50
n51	integer	Node number 51
n52	integer	Node number 52
n53	integer	Node number 53
n54	integer	Node number 54
n55	integer	Node number 55
n56	integer	Node number 56
n57	integer	Node number 57
n58	integer	Node number 58
n59	integer	Node number 59
n6	integer	Node number 6
n60	integer	Node number 60
n61	integer	Node number 61
n62	integer	Node number 62
n63	integer	Node number 63
n64	integer	Node number 64

n7	integer	Node number 7
n8	integer	Node number 8
n9	integer	Node number 9
nodes	integer	Number of nodes solid has (read only)
ns1	integer	Scalar Node number 1
ns2	integer	Scalar Node number 2
ns3	integer	Scalar Node number 3
ns4	integer	Scalar Node number 4
ns5	integer	Scalar Node number 5
ns6	integer	Scalar Node number 6
ns7	integer	Scalar Node number 7
ns8	integer	Scalar Node number 8
ortho	logical	If _ORTHO option is set. Can be true or false
p21	logical	If _P21 option is set. Can be true or false
p40	logical	If _P40 option is set. Can be true or false
pid	integer	Part number
t15	logical	If _T15 option is set. Can be true or false
t20	logical	If _T20 option is set. Can be true or false
tet4totet10	logical	If _TET4TOTET10 option is set. Can be true or false
transparency	integer	The transparency of the solid (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Solid class allows you to create, modify, edit and manipulate solid cards. See the documentation below for more details.

Constructor

new Solid(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, n5 (optional)*[integer]*, n6 (optional)*[integer]*, n7 (optional)*[integer]*, n8 (optional)*[integer]*, n9 (optional)*[integer]*, n10 (optional)*[integer]*, n11 (optional)*[integer]*, n12 (optional)*[integer]*, n13 (optional)*[integer]*, n14 (optional)*[integer]*, n15 (optional)*[integer]*, n16 (optional)*[integer]*, n17 (optional)*[integer]*, n18 (optional)*[integer]*, n19 (optional)*[integer]*, n20 (optional)*[integer]*, n21 (optional)*[integer]*, n22 (optional)*[integer]*, n23 (optional)*[integer]*, n24 (optional)*[integer]*, n25 (optional)*[integer]*, n26 (optional)*[integer]*, n27 (optional)*[integer]*)

Description

Create a new Solid object. Use either 4, 6, 8 or 10 nodes when creating a new solid. If you are creating a 4 noded solid either only give 4 nodes or give 8 nodes but make nodes 4 to 8 the same number. If you are creating a 6 noded solid either only give 6 nodes or give 8 nodes but make nodes 5 and 6 the same number and nodes 7 and 8 the same number.

Arguments

Name	Type	Description

Model	Model	Model that solid will be created in
eid	integer	Solid number
pid	integer	Part number
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
n4	integer	Node number 4
n5 (optional)	integer	Node number 5
n6 (optional)	integer	Node number 6
n7 (optional)	integer	Node number 7
n8 (optional)	integer	Node number 8
n9 (optional)	integer	Node number 9
n10 (optional)	integer	Node number 10
n11 (optional)	integer	Node number 11
n12 (optional)	integer	Node number 12
n13 (optional)	integer	Node number 13
n14 (optional)	integer	Node number 14
n15 (optional)	integer	Node number 15
n16 (optional)	integer	Node number 16
n17 (optional)	integer	Node number 17
n18 (optional)	integer	Node number 18
n19 (optional)	integer	Node number 19
n20 (optional)	integer	Node number 20
n21 (optional)	integer	Node number 21
n22 (optional)	integer	Node number 22
n23 (optional)	integer	Node number 23
n24 (optional)	integer	Node number 24
n25 (optional)	integer	Node number 25
n26 (optional)	integer	Node number 26
n27 (optional)	integer	Node number 27

Return type

Solid object

Example

To create a new solid in model m with label 100, part 10 and nodes 1, 2, 3, 4, 5, 6, 7, 8:

```
var s = new Solid(m, 100, 10, 1, 2, 3, 4, 5, 6, 7, 8);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a solid.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the solid

Return type

No return value

Example

To associate comment c to the solid s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the solid

Arguments

No arguments

Return type

No return value

Example

To blank solid s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the solids in model m:

```
Solid.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged solids will be blanked in
flag	Flag	Flag set on the solids that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the solids in model m flagged with f:

```
Solid.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the solid is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if solid s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse solid s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the solid.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the solid

Return type

No return value

Example

To clear flag f for solid s:

```
s.ClearFlag(f);
```

CoordsToIsoparametric(Model*[Model]*, x*[real]*, y*[real]*, z*[real]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*) [static]

Description

Calculates the isoparametric coordinates for a point on 3 or 4 noded segment

Arguments

Name	Type	Description
Model	Model	Model designated model
x	real	X coordinate of point
y	real	Y coordinate of point
z	real	Z coordinate of point
n1	integer	node 1 of segment
n2	integer	node 2 of segment
n3	integer	node 3 of segment
n4	integer	node 4 of segment

Return type

Array containing s and t isoparametric coordinates and the distance the point is from the segment If it is not possible to calculate the isoparametric coordinates null is returned.

Example

To calculate the isoparametric coordinates of point (100, 100, 20) on segment defined by nodes 11,12,13,14:

```
var isocoords = Solid.CoordsToIsoparametric(100, 100, 20, 11, 12, 13, 14);
```

Copy(range (optional)*[boolean]*)

Description

Copies the solid.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Solid object

Example

To copy solid s into solid z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a solid.

Arguments

Name	Type	Description
Model	Model	Model that the solid will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Solid object (or null if not made)

Example

To start creating a solid in model m:

```
var s = Solid.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a solid.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be detached from the solid

Return type

No return value

Example

To detach comment c from the solid s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit solid s:

```
s.Edit();
```

ElemCut(Database cross section label*[integer]*)

Description

Returns coordinates of the intersections between a solid and a database cross section.

Arguments

Name	Type	Description
Database cross section label	integer	The label of the database cross section.

Return type

Object with the following properties:

Name	Type	Description
face1	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 1. Null if no cut on this face.
face2	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 2. Null if no cut on this face.
face3	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 3. Null if no cut on this face.
face4	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 4. Null if no cut on this face.
face5	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 5. Null if no cut on this face.

face6	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 6. Null if no cut on this face.

Example

To see if the database cross section 200 cuts solid s and at which points it cuts face 3 of the solid:

```
var data = s.ElemCut(200);
var face = data.face3;
if(face)
{
   var point1_x = face[0];
   var point1_y = face[1];
   var point1_z = face[2];
   var point2_x = face[3];
   var point2_y = face[4];
   var point2_z = face[5];
}
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for solid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for solid s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for solid.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the solid colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the solid.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing solid s:

```
var colour = s.ExtractColour();
```

FindSolidEnd() [static] [deprecated]

This function is deprecated in version 16.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Tidy memory allocation incurred by function which finds solids within a box. Now replaced by model member function Model.FindElemEnd()

Arguments

No arguments

Return type

No return value

Example

```
Solid.FindSolidEnd();
```

FindSolidInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, hflag (optional)*[integer]*) [static]

Description

Returns an array of Solid objects for the solids within a box. This requires a previous (outside loop) call to function FindSolidInit(m) or m.FindElemInit() where the process is initialized for flagged solids in the model (typically all solids) and m.FindElemEnd() to close the process. Please note this function provides a list of all solids that could potentially be in the box (using computationally cheap bounding box comparison) it is not a rigorous test of whether the solid is actually in the box. See also Solid.FindSolidInit() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model designated model
xmin	real	Minimum bound in global x
xmax	real	Maximum bound in global x
ymin	real	Minimum bound in global y
ymax	real	Maximum bound in global y
zmin	real	Minimum bound in global z
zmax	real	Maximum bound in global z
hflag (optional)	integer	Optional flag to restrict solids considered

Return type

Array of Solid objects

Example

To get an array of Solid objects for solids in model m within defined box

```
Solid.FindSolidInit(m);
```

or clear model flag and flag elements of interest and

```
m.FindElemInit(flag);

{

    //loop in which boxes are formed and tested

    //find solids both in box and flagged with hflag

    var s = Solid.FindSolidInBox(m, xmin, xmax, ymin, ymax, zmin, zmax, hflag);

    if(s.length) ...

}

m.FindElemEnd();
```

FindSolidInit(Model*[Model]*, flag (optional)*[Flag]*) [static]

Description

Initialize setup so that all flagged solids in model can be tested to see if they are within box. See also Solid.FindSolidInBox() See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model in which solids have been flagged
flag (optional)	Flag	Optional flag that has been set on the solids, if unsupplied all solids considered

Return type

No return value

Example

To initialize find setup for all solids in model m:

```
Solid.FindSolidInit(m);
```

First(Model*[Model]*) [static]

Description

Returns the first solid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first solid in

Return type

Solid object (or null if there are no solids in the model).

Example

To get the first solid in model m:

```
var s = Solid.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free solid label in the model. Also see Solid.LastFreeLabel(), Solid.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free solid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Solid label.

Example

To get the first free solid label in model m:

```
var label = Solid.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the solids in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be flagged in
flag	Flag	Flag to set on the solids

Return type

No return value

Example

To flag all of the solids with flag f in model m:

```
Solid.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the solid is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the solid

Return type

true if flagged, false if not.

Example

To check if solid s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each solid in the model.
Note that ForEach has been designed to make looping over solids as fast as possible and so has some limitations. Firstly, a single temporary Solid object is created and on each function call it is updated with the current solid data. This means that you should not try to store the Solid object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new solids inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all solids are in
func	function	Function to call for each solid
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the solids in model m:

```
Solid.ForEach(m, test);
function test(s)
{
// s is Solid object
}
```

To call function test for all of the solids in model m with optional object:

```
var data = { x:0, y:0 };
Solid.ForEach(m, test, data);
function test(s, extra)
{
// s is Solid object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Solid objects for all of the solids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get solids from

Return type

Array of Solid objects

Example

To make an array of Solid objects for all of the solids in model m

```
var s = Solid.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a solid.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the solid s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Solid objects for all of the flagged solids in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get solids from
flag	Flag	Flag set on the solids that you want to retrieve

Return type

Array of Solid objects

Example

To make an array of Solid objects for all of the solids in model m flagged with f

```
var s = Solid.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Solid object for a solid ID.

Arguments

Name	Type	Description
Model	Model	Model to find the solid in
number	integer	number of the solid you want the Solid object for

Return type

Solid object (or null if solid does not exist).

Example

To get the Solid object for solid 100 in model m

```
var s = Solid.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Solid property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Solid.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	solid property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Solid property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Solid property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this solid (*SOLID, *SOLID_SCALAR or *SOLID_SCALAR_VALUE). **Note that a carriage return is not added**. See also Solid.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for solid s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the solid. **Note that a carriage return is not added**. See also Solid.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for solid s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last solid in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last solid in

Return type

Solid object (or null if there are no solids in the model).

Example

To get the last solid in model m:

```
var s = Solid.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free solid label in the model. Also see Solid.FirstFreeLabel(), Solid.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free solid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Solid label.

Example

To get the last free solid label in model m:

```
var label = Solid.LastFreeLabel(m);
```

Next()

Description

Returns the next solid in the model.

Arguments

No arguments

Return type

Solid object (or null if there are no more solids in the model).

Example

To get the solid in model m after solid s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) solid label in the model. Also see Solid.FirstFreeLabel(), Solid.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free solid label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Solid label.

Example

To get the next free solid label in model m:

```
var label = Solid.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a solid.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only solids from that model can be picked. If the argument is a Flag then only solids that are flagged with *limit* can be selected. If omitted, or null, any solids from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Solid object (or null if not picked)

Example

To pick a solid from model m giving the prompt 'Pick solid from screen':

```
var s = Solid.Pick('Pick solid from screen', m);
```

Previous()

Description

Returns the previous solid in the model.

Arguments

No arguments

Return type

Solid object (or null if there are no more solids in the model).

Example

To get the solid in model m before solid s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the solids in model m, from 1000000:

```
Solid.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged solids will be renumbered in
flag	Flag	Flag set on the solids that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the solids in model m flagged with f, from 1000000:

```
Solid.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select solids using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting solids
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only solids from that model can be selected. If the argument is a Flag then only solids that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any solids can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of solids selected or null if menu cancelled

Example

To select solids from model m, flagging those selected with flag f, giving the prompt ’Select solids’:

```
Solid.Select(f, ’Select solids’, m);
```

To select solids, flagging those selected with flag f but limiting selection to solids flagged with flag l, giving the prompt ’Select solids’:

```
Solid.Select(f, ’Select solids’, l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the solid.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the solid

Return type

No return value

Example

To set flag f for solid s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the solid. The solid will be sketched until you either call Solid.Unsketch(), Solid.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the solid is sketched. If omitted redraw is true. If you want to sketch several solids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch solid s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged solids in the model. The solids will be sketched until you either call Solid.Unsketch(), Solid.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged solids will be sketched in
flag	Flag	Flag set on the solids that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the solids are sketched. If omitted redraw is true. If you want to sketch flagged solids several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all solids flagged with flag in model m:

```
Solid.SketchFlagged(m, flag);
```

TetCollapse()

Description

Calculates the tetrahedral collapse for the solid

Arguments

No arguments

Return type

real

Example

To calculate the tet collapse for solid s:

```
var tet collapse = s.TetCollapse();
```

TiedNodeCheck(Contact label*[integer]*, Flag*[Flag]*, Option1*[integer]*, Option2*[integer]*)

Description

Checks if nodes of solid are tied by contact or directly attached (non-zero option1).

Arguments

Name	Type	Description
Contact label	integer	The label of the tied contact. If zero the tied contact is found for the solid by reverse lookup.
Flag	Flag	flag bit
Option1	integer	Directly tied node (logical OR) 0:NONE 1:NRB/C_EXNO 2:BEAM 4:SHELL 8:SOLID 16:TSHELL
Option2	integer	0:No action 1:report error if directly attached node (acc. option1) also captured by contact

Return type

string

Example

To check if all nodes of solid h are tied by contact 200 or attach directly to constraint or shell:

```
var message = h.TiedNodeCheck(200, flag, 1|4, 1)
```

Timestep()

Description

Calculates the timestep for the solid

Arguments

No arguments

Return type

real

Example

To calculate the timestep for solid s:

```
var timestep = s.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of solids in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing solids should be counted. If false or omitted referenced but undefined solids will also be included in the total.

Return type

number of solids

Example

To get the total number of solids in model m:

```
var total = Solid.Total(m);
```

Unblank()

Description

Unblanks the solid

Arguments

No arguments

Return type

No return value

Example

To unblank solid s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the solids in model m:

```
Solid.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged solids will be unblanked in
flag	Flag	Flag set on the solids that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the solids in model m flagged with f:

```
Solid.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all solids will be unset in

flag	Flag	Flag to unset on the solids

Return type

No return value

Example

To unset the flag f on all the solids in model m:

```
Solid.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the solid.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the solid is unsketched. If omitted redraw is true. If you want to unsketch several solids and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch solid s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all solids.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the solids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all solids in model m:

```
Solid.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged solids in the model.

Arguments

Name	Type	Description
Model	Model	Model that all solids will be unsketched in
flag	Flag	Flag set on the solids that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the solids are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all solids flagged with flag in model m:

```
Solid.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Solid object.

Example

To check if Solid property s.example is a parameter by using the Solid.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Volume()

Description

Calculates the volume for the solid

Arguments

No arguments

Return type

real

Example

To calculate the volume for solid s:

```
var volume = s.Volume();
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for solid. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for solid s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this solid.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for solid s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the solid data in keyword format. Note that this contains the keyword header and the keyword cards. See also Solid.Keyword() and Solid.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for solid s in keyword format

```
var str = s.toString();
```

Sph class

The Sph class gives you access to Element SPH cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

- toString()

Sph properties

Name	Type	Description
colour	Colour	The colour of the sph element.
exists	logical	true if sph exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the sph is in.
mass	real	Mass value.
model	integer	The Model number that the sph is in.
nid	integer	Node ID.
pid	integer	Part ID to which this element belongs.
transparency	integer	The transparency of the sph (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Sph class allows you to create, modify, edit and manipulate SPH cards. See the documentation below for more details.

Constructor

new Sph(Model*[Model]*, nid*[integer]*, pid*[integer]*, mass*[real]*)

Description

Create a new object.

Arguments

Name	Type	Description
Model	Model	Model that sph will be created in
nid	integer	Node ID and Element ID are the same for the SPH option.
pid	integer	Part ID to which this element belongs.
mass	real	Mass value.

Return type

Sph object

Example

To create a new sph element in model m with nid = 100, pid = 400, mass = 0.9:

```
var s = new Sph(m, 100, 400, 0.9);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a sph.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the sph

Return type

No return value

Example

To associate comment c to the sph s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the sph

Arguments

No arguments

Return type

No return value

Example

To blank sph s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sphs will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sphs in model m:

```
Sph.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sphs will be blanked in
flag	Flag	Flag set on the sphs that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sphs in model m flagged with f:

```
Sph.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the sph is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if sph s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse sph s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the sph.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the sph

Return type

No return value

Example

To clear flag f for sph s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the sph.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Sph object

Example

To copy sph s into sph z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an sph.

Arguments

Name	Type	Description
Model	Model	Model that the sph will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Sph object (or null if not made)

Example

To start creating an sph in model m:

```
var s = Sph.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a sph.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the sph

Return type

No return value

Example

To detach comment c from the sph s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit sph s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for sph. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for sph s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for sph.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the sph colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the sph.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing sph s:

```
var colour = s.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first sph in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first sph in

Return type

Sph object (or null if there are no sphs in the model).

Example

To get the first sph in model m:

```
var s = Sph.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free sph label in the model. Also see Sph.LastFreeLabel(), Sph.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free sph label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Sph label.

Example

To get the first free sph label in model m:

```
var label = Sph.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the sphs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all sphs will be flagged in
flag	Flag	Flag to set on the sphs

Return type

No return value

Example

To flag all of the sphs with flag f in model m:

```
Sph.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the sph is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the sph

Return type

true if flagged, false if not.

Example

To check if sph s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each sph in the model.
Note that ForEach has been designed to make looping over sphs as fast as possible and so has some limitations. Firstly, a single temporary Sph object is created and on each function call it is updated with the current sph data. This means that you should not try to store the Sph object for later use (e.g. in an array) as it is temporary. Secondly, you cannot create new sphs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all sphs are in
func	function	Function to call for each sph
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the sphs in model m:

```
Sph.ForEach(m, test);
function test(s)
{
// s is Sph object
}
```

To call function test for all of the sphs in model m with optional object:

```
var data = { x:0, y:0 };
Sph.ForEach(m, test, data);
function test(s, extra)
{
// s is Sph object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Sph objects for all of the sphs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sphs from

Return type

Array of Sph objects

Example

To make an array of Sph objects for all of the sphs in model m

```
var s = Sph.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a sph.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the sph s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Sph objects for all of the flagged sphs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sphs from
flag	Flag	Flag set on the sphs that you want to retrieve

Return type

Array of Sph objects

Example

To make an array of Sph objects for all of the sphs in model m flagged with f

```
var s = Sph.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Sph object for a sph ID.

Arguments

Name	Type	Description
Model	Model	Model to find the sph in
number	integer	number of the sph you want the Sph object for

Return type

Sph object (or null if sph does not exist).

Example

To get the Sph object for sph 100 in model m

```
var s = Sph.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Sph property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Sph.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	sph property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Sph property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Sph property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this sph (*ELEMENT_SPH) **Note that a carriage return is not added**. See also Sph.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for sph s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the sph. **Note that a carriage return is not added**. See also Sph.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for sph s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last sph in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last sph in

Return type

Sph object (or null if there are no sphs in the model).

Example

To get the last sph in model m:

```
var s = Sph.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free sph label in the model. Also see Sph.FirstFreeLabel(), Sph.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free sph label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Sph label.

Example

To get the last free sph label in model m:

```
var label = Sph.LastFreeLabel(m);
```

Next()

Description

Returns the next sph in the model.

Arguments

No arguments

Return type

Sph object (or null if there are no more sphs in the model).

Example

To get the sph in model m after sph s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) sph label in the model. Also see Sph.FirstFreeLabel(), Sph.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free sph label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Sph label.

Example

To get the next free sph label in model m:

```
var label = Sph.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a sph.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sphs from that model can be picked. If the argument is a Flag then only sphs that are flagged with *limit* can be selected. If omitted, or null, any sphs from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Sph object (or null if not picked)

Example

To pick a sph from model m giving the prompt 'Pick sph from screen':

```
var s = Sph.Pick('Pick sph from screen', m);
```

Previous()

Description

Returns the previous sph in the model.

Arguments

No arguments

Return type

Sph object (or null if there are no more sphs in the model).

Example

To get the sph in model m before sph s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sphs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sphs in model m, from 1000000:

```
Sph.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sphs will be renumbered in
flag	Flag	Flag set on the sphs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sphs in model m flagged with f, from 1000000:

```
Sph.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select sphs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting sphs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sphs from that model can be selected. If the argument is a Flag then only sphs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any sphs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of sphs selected or null if menu cancelled

Example

To select sphs from model m, flagging those selected with flag f, giving the prompt 'Select sphs':

```
Sph.Select(f, 'Select sphs', m);
```

To select sphs, flagging those selected with flag f but limiting selection to sphs flagged with flag l, giving the prompt 'Select sphs':

```
Sph.Select(f, 'Select sphs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the sph.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the sph

Return type

No return value

Example

To set flag f for sph s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the sph. The sph will be sketched until you either call Sph.Unsketch(), Sph.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the sph is sketched. If omitted redraw is true. If you want to sketch several sphs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch sph s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged sphs in the model. The sphs will be sketched until you either call Sph.Unsketch(), Sph.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sphs will be sketched in
flag	Flag	Flag set on the sphs that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the sphs are sketched. If omitted redraw is true. If you want to sketch flagged sphs several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all sphs flagged with flag in model m:

```
Sph.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing sphs should be counted. If false or omitted referenced but undefined sphs will also be included in the total.

Return type

number of sphs

Example

To get the total number of sphs in model m:

```
var total = Sph.Total(m);
```

Unblank()

Description

Unblanks the sph

Arguments

No arguments

Return type

No return value

Example

To unblank sph s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the sphs in the model.

Arguments

Name	Type	Description

Model	Model	Model that all sphs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sphs in model m:

```
Sph.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged sphs will be unblanked in
flag	Flag	Flag set on the sphs that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sphs in model m flagged with f:

```
Sph.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all sphs will be unset in
flag	Flag	Flag to unset on the sphs

Return type

No return value

Example

To unset the flag f on all the sphs in model m:

```
Sph.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the sph.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the sph is unsketched. If omitted redraw is true. If you want to unsketch several sphs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch sph s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all sphs.

Arguments

Name	Type	Description
Model	Model	Model that all sphs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the sphs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sphs in model m:

```
Sph.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged sphs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sphs will be unsketched in
flag	Flag	Flag set on the sphs that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the sphs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sphs flagged with flag in model m:

```
Sph.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Sph object.

Example

To check if Sph property s.example is a parameter by using the Sph.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for sph. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for sph s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this sph.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for sph s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the sph data in keyword format. Note that this contains the keyword header and the keyword cards. See also Sph.Keyword() and Sph.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for sph s in keyword format

```
var str = s.toString();
```

Tshell class

The Tshell class gives you access to thick shell cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- FindTshellInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, hflag (optional)*[integer]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- ElemCut(Database cross section label*[integer]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetCompositeData(ipt*[integer]*)
- GetNodeIDs()
- GetNodes()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveCompositeData(ipt*[integer]*)

- SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Timestep()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Tshell properties

Name	Type	Description
beta	logical	If BETA option is set.
beta_angle	real	Angle for BETA option.
colour	Colour	The colour of the thick shell
composite	logical	If COMPOSITE option is set. Can be true or false
eid	integer	Tshell number. Also see the label property which is an alternative name for this.
exists	logical	true if thick shell exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the thick shell is in.
label	integer	Tshell number. Also see the eid property which is an alternative name for this.
model	integer	The Model number that the thick shell is in.
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
n4	integer	Node number 4
n5	integer	Node number 5
n6	integer	Node number 6
n7	integer	Node number 7
n8	integer	Node number 8
nip	logical	Number of integration points for composite thick shell
nodes	integer	Number of nodes thick shell has (read only)
pid	integer	Part number
transparency	integer	The transparency of the thick shell (0-100) 0% is opaque, 100% is transparent.

Detailed Description

The Tshell class allows you to create, modify, edit and manipulate thich shell cards. See the documentation below for more details.

Constructor

new Tshell(Model*[Model]*, eid*[integer]*, pid*[integer]*, n1*[integer]*, n2*[integer]*, n3*[integer]*, n4*[integer]*, n5*[integer]*, n6*[integer]*, n7 (optional)*[integer]*, n8 (optional)*[integer]*)

Description

Create a new Tshell object. Use either 6 or 8 nodes when creating a new thick shell.

Arguments

Name	Type	Description
Model	Model	Model that thick shell will be created in
eid	integer	Tshell number
pid	integer	Part number
n1	integer	Node number 1
n2	integer	Node number 2
n3	integer	Node number 3
n4	integer	Node number 4
n5	integer	Node number 5
n6	integer	Node number 6
n7 (optional)	integer	Node number 7
n8 (optional)	integer	Node number 8

Return type

Tshell object

Example

To create a new thick shell in model m with label 100, part 10 and nodes 1, 2, 3, 4, 5, 6, 7, 8:

```
var t = new Tshell(m, 100, 10, 1, 2, 3, 4, 5, 6, 7, 8);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a thick shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the thick shell

Return type

No return value

Example

To associate comment c to the thick shell t:

```
t.AssociateComment(c);
```

Blank()

Description

Blanks the thick shell

Arguments

No arguments

Return type

No return value

Example

To blank thick shell t:

```
t.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the thick shells in model m:

```
Tshell.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged thick shells will be blanked in
flag	Flag	Flag set on the thick shells that you want to blank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the thick shells in model m flagged with f:

```
Tshell.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the thick shell is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if thick shell t is blanked:

```
if (t.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse thick shell t:

```
t.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the thick shell.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the thick shell

Return type

No return value

Example

To clear flag f for thick shell t:

```
t.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the thick shell.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Tshell object

Example

To copy thick shell t into thick shell z:

```
var z = t.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a thick shell.

Arguments

Name	Type	Description
Model	Model	Model that the thick shell will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Tshell object (or null if not made)

Example

To start creating a thick shell in model m:

```
var t = Tshell.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a thick shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the thick shell

Return type

No return value

Example

To detach comment c from the thick shell t:

```
t.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit thick shell t:

```
t.Edit();
```

ElemCut(Database cross section label*[integer]*)

Description

Returns coordinates of the intersections between a thick shell and a database cross section.

Arguments

Name	Type	Description
Database cross section label	integer	The label of the database cross section.

Return type

Object with the following properties:

Name	Type	Description
face1	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 1. Null if no cut on this face.

face2	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 2. Null if no cut on this face.
face3	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 3. Null if no cut on this face.
face4	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 4. Null if no cut on this face.
face5	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 5. Null if no cut on this face.
face6	Array of reals	An array containing the x1,y1,z1,x2,y2,z2 coordinates of the cut line on the face 6. Null if no cut on this face.

Example

To see if the database cross section 200 cuts thick shell s and at which points it cuts face 3 of the thick shell:

```
var data = s.ElemCut(200);
var face = data.face3;
if(face)
{
    var point1_x = face[0];
    var point1_y = face[1];
    var point1_z = face[2];
    var point2_x = face[3];
    var point2_y = face[4];
    var point2_z = face[5];
}
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for thick shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for thick shell t:

```
t.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for thick shell.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the thick shell colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the thick shell.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing thick shell t:

```
var colour = t.ExtractColour();
```

FindTshellInBox(Model*[Model]*, xmin*[real]*, xmax*[real]*, ymin*[real]*, ymax*[real]*, zmin*[real]*, zmax*[real]*, hflag (optional)*[integer]*) [static]

Description

Returns an array of Tshell objects for the thick shells within a box. This requires a previous (outside loop) call to function m.FindElemInit() where the process is initialized for flagged thick shells in the model and m.FindElemEnd() to close the process. Please note this function provides a list of all thick shells that could potentially be in the box (using computationally cheap bounding box comparison) it is not a rigorous test of whether the thick shellis actually in the box. See also Model.FindElemInit() See also Model.FindElemEnd()

Arguments

Name	Type	Description
Model	Model	Model designated model
xmin	real	Minimum bound in global x
xmax	real	Maximum bound in global x
ymin	real	Minimum bound in global y
ymax	real	Maximum bound in global y
zmin	real	Minimum bound in global z
zmax	real	Maximum bound in global z
hflag (optional)	integer	Optional flag to restrict thick shells considered

Return type

Array of Tshell objects

Example

To get an array of Tshell objects for flagged thick shells in model m within defined box

```
m.FindElemInit(flag);

{

    //loop in which boxes are formed and tested

    //find thick shells both in box and flagged with hflag

    var s = Tshell.FindTshellInBox(m, xmin, xmax, ymin, ymax, zmin, zmax,
hflag);

    if(s.length) ...

}

m.FindElemEnd();
```

First(Model*[Model]*) [static]

Description

Returns the first thick shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first thick shell in

Return type

Tshell object (or null if there are no thick shells in the model).

Example

To get the first thick shell in model m:

```
var t = Tshell.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free thick shell label in the model. Also see Tshell.LastFreeLabel(), Tshell.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free thick shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Tshell label.

Example

To get the first free thick shell label in model m:

```
var label = Tshell.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the thick shells in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be flagged in
flag	Flag	Flag to set on the thick shells

Return type

No return value

Example

To flag all of the thick shells with flag f in model m:

```
Tshell.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the thick shell is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the thick shell

Return type

true if flagged, false if not.

Example

To check if thick shell t has flag f set on it:

```
if (t.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each thick shell in the model.
Note that ForEach has been designed to make looping over thick shells as fast as possible and so has some limitations.
Firstly, a single temporary Tshell object is created and on each function call it is updated with the current thick shell data. This means that you should not try to store the Tshell object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new thick shells inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells are in
func	function	Function to call for each thick shell
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the thick shells in model m:

```
Tshell.ForEach(m, test);
function test(t)
{
// t is Tshell object
}
```

To call function test for all of the thick shells in model m with optional object:

```
var data = { x:0, y:0 };
Tshell.ForEach(m, test, data);
function test(t, extra)
{
// t is Tshell object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Tshell objects for all of the thick shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get thick shells from

Return type

Array of Tshell objects

Example

To make an array of Tshell objects for all of the thick shells in model m

```
var t = Tshell.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a thick shell.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the thick shell t:

```
var comm_array = t.GetComments();
```

GetCompositeData(ipt*[integer]*)

Description

Returns the composite data for an integration point in *ELEMENT_TSHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want the data for. **Note that integration points start at 0, not 1**.

Return type

An array of numbers containing the material id, thickness and beta angle.

Example

To get the composite data for the 3rd integration point for thick shell t:

```
if (t.composite && s.nip >= 3)
{
    var ipt_data = t.GetCompositeData(2);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Tshell objects for all of the flagged thick shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get thick shells from
flag	Flag	Flag set on the thick shells that you want to retrieve

Return type

Array of Tshell objects

Example

To make an array of Tshell objects for all of the thick shells in model m flagged with f

```
var t = Tshell.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Tshell object for a thick shell ID.

Arguments

Name	Type	Description
Model	Model	Model to find the thick shell in
number	integer	number of the thick shell you want the Tshell object for

Return type

Tshell object (or null if thick shell does not exist).

Example

To get the Tshell object for thick shell 100 in model m

```
var t = Tshell.GetFromID(m, 100);
```

GetNodeIDs()

Description

Returns the labels of the nodes on the thick shell as an array. See also Tshell.GetNodes()

Arguments

No arguments

Return type

Array of node labels (integers)

Example

To return the node labels of thick shell t as an array

```
var nodes = t.GetNodeIDs();
```

GetNodes()

Description

Returns the nodes on the thick shell as an array of Node objects. See also Tshell.GetNodeIDs()

Arguments

No arguments

Return type

Array of Node objects

Example

To return the nodes of thick shell t as an array

```
var nodes = t.GetNodes();
```

GetParameter(prop*[string]*)

Description

Checks if a Tshell property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Tshell.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	thick shell property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Tshell property t.example is a parameter:

```
Options.property_parameter_names = true;
if (t.GetParameter(t.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Tshell property t.example is a parameter by using the GetParameter method:

```
if (t.ViewParameters().GetParameter(t.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this thick shell (*ELEMENT_TSHELL or *ELEMENT_TSHELL_COMPOSITE). **Note that a carriage return is not added**. See also Tshell.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for thick shell t:

```
var key = t.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the thick shell. **Note that a carriage return is not added**. See also Tshell.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for thick shell t:

```
var cards = t.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last thick shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last thick shell in

Return type

Tshell object (or null if there are no thick shells in the model).

Example

To get the last thick shell in model m:

```
var t = Tshell.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free thick shell label in the model. Also see Tshell.FirstFreeLabel(), Tshell.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free thick shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Tshell label.

Example

To get the last free thick shell label in model m:

```
var label = Tshell.LastFreeLabel(m);
```

Next()

Description

Returns the next thick shell in the model.

Arguments

No arguments

Return type

Tshell object (or null if there are no more thick shells in the model).

Example

To get the thick shell in model m after thick shell t:

```
var t = t.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) thick shell label in the model. Also see Tshell.FirstFreeLabel(), Tshell.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free thick shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Tshell label.

Example

To get the next free thick shell label in model m:

```
var label = Tshell.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a thick shell.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only thick shells from that model can be picked. If the argument is a Flag then only thick shells that are flagged with *limit* can be selected. If omitted, or null, any thick shells from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Tshell object (or null if not picked)

Example

To pick a thick shell from model m giving the prompt 'Pick thick shell from screen':

```
var t = Tshell.Pick('Pick thick shell from screen', m);
```

Previous()

Description

Returns the previous thick shell in the model.

Arguments

No arguments

Return type

Tshell object (or null if there are no more thick shells in the model).

Example

To get the thick shell in model m before thick shell t:

```
var t = t.Previous();
```

RemoveCompositeData(ipt*[integer]*)

Description

Removes the composite data for an integration point in *ELEMENT_TSHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to remove. **Note that integration points start at 0, not 1**.

Return type

No return value.

Example

To remove the composite data for the 3rd integration point for thick shell t:

```
t.RemoveCompositeData(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the thick shells in model m, from 1000000:

```
Tshell.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged thick shells will be renumbered in
flag	Flag	Flag set on the thick shells that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the thick shells in model m flagged with f, from 1000000:

```
Tshell.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select thick shells using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting thick shells
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only thick shells from that model can be selected. If the argument is a Flag then only thick shells that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any thick shells can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of thick shells selected or null if menu cancelled

Example

To select thick shells from model m, flagging those selected with flag f, giving the prompt ’Select thick shells’:

```
Tshell.Select(f, ’Select thick shells’, m);
```

To select thick shells, flagging those selected with flag f but limiting selection to thick shells flagged with flag l, giving the prompt ’Select thick shells’:

```
Tshell.Select(f, ’Select thick shells’, l);
```

SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*)

Description

Sets the composite data for an integration point in *ELEMENT_TSHELL_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to set the data for. **Note that integration points start at 0, not 1**.
mid	integer	Material ID for the integration point.
thick	real	Thickness of the integration point.
beta	real	Material angle of the integration point.

Return type

No return value.

Example

To set the composite data for the 3rd integration point to mat 1, thickness 0.5 and angle 45, for thick shell t:

```
t.SetCompositeData(2, 1, 0.5, 45);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the thick shell.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the thick shell

Return type

No return value

Example

To set flag f for thick shell t:

```
t.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the thick shell. The thick shell will be sketched until you either call Tshell.Unsketch(), Tshell.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the thick shell is sketched. If omitted redraw is true. If you want to sketch several thick shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch thick shell t:

```
t.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged thick shells in the model. The thick shells will be sketched until you either call Tshell.Unsketch(), Tshell.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged thick shells will be sketched in
flag	Flag	Flag set on the thick shells that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the thick shells are sketched. If omitted redraw is true. If you want to sketch flagged thick shells several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all thick shells flagged with flag in model m:

```
Tshell.SketchFlagged(m, flag);
```

Timestep()

Description

Calculates the timestep for the thick shell

Arguments

No arguments

Return type

real

Example

To calculate the timestep for thick shell t:

```
var timestep = t.Timestep();
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing thick shells should be counted. If false or omitted referenced but undefined thick shells will also be included in the total.

Return type

number of thick shells

Example

To get the total number of thick shells in model m:

```
var total = Tshell.Total(m);
```

Unblank()

Description

Unblanks the thick shell

Arguments

No arguments

Return type

No return value

Example

To unblank thick shell t:

```
t.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the thick shells in model m:

```
Tshell.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged thick shells will be unblanked in
flag	Flag	Flag set on the thick shells that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the thick shells in model m flagged with f:

```
Tshell.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all thick shells will be unset in
flag	Flag	Flag to unset on the thick shells

Return type

No return value

Example

To unset the flag f on all the thick shells in model m:

```
Tshell.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the thick shell.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the thick shell is unsketched. If omitted redraw is true. If you want to unsketch several thick shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch thick shell t:

```
t.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all thick shells.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the thick shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all thick shells in model m:

```
Tshell.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged thick shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all thick shells will be unsketched in
flag	Flag	Flag set on the thick shells that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the thick shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all thick shells flagged with flag in model m:

```
Tshell.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Tshell object.

Example

To check if Tshell property t.example is a parameter by using the Tshell.GetParameter() method:

```
if (t.ViewParameters().GetParameter(t.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for thick shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for thick shell t:

```
t.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this thick shell.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for thick shell t:

```
var xrefs = t.Xrefs();
```

---

toString()

Description

Creates a string containing the thick shell data in keyword format. Note that this contains the keyword header and the keyword cards. See also Tshell.Keyword() and Tshell.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for thick shell t in keyword format

```
var str = t.toString();
```

Hourglass class

The Hourglass class gives you access to hourglass cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Hourglass properties

Name	Type	Description
exists	logical	true if hourglass exists, false if referred to but not defined. (read only)
hgid	integer or string	Hourglass number or character label. Also see the label property which is an alternative name for this.
ibq	integer	Bulk viscosity type

ihq	integer	Hourglass control type
include	integer	The Include file number that the hourglass is in.
label	integer or string	Hourglass number or character label. Also see the hgid property which is an alternative name for this.
model	integer	The Model number that the hourglass is in.
q1	real	Quadratic bulk viscosity coefficient
q2	real	Linear bulk viscosity coefficient
qb	real	Coefficient for shell bending
qm	real	Hourglass coefficient
qw	real	Coefficient for shell warping
title	string	Title for hourglass
vdc	real	Viscous damping coefficient

Detailed Description

The Hourglass class allows you to create, modify, edit and manipulate hourglass cards. See the documentation below for more details.

Constructor

new Hourglass(Model*[Model]*, hgid*[integer or string]*, title (optional)*[string]*)

Description

Create a new Hourglass object.

Arguments

Name	Type	Description
Model	Model	Model that hourglass will be created in
hgid	integer or string	Hourglass number or character label
title (optional)	string	Title for the hourglass

Return type

Hourglass object

Example

To create a new hourglass in model m with label 100:

```
var h = new Hourglass(m, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a hourglass.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be attached to the hourglass

Return type

No return value

Example

To associate comment c to the hourglass n:

```
n.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse hourglass n:

```
n.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the hourglass.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the hourglass

Return type

No return value

Example

To clear flag f for hourglass n:

```
n.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the hourglass.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Hourglass object

Example

To copy hourglass n into hourglass z:

```
var z = n.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a hourglass.

Arguments

Name	Type	Description
Model	Model	Model that the hourglass will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Hourglass object (or null if not made)

Example

To start creating a hourglass in model m:

```
var h = Hourglass.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a hourglass.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the hourglass

Return type

No return value

Example

To detach comment c from the hourglass n:

```
n.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit hourglass n:

```
n.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for hourglass. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for hourglass n:

```
n.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first hourglass in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first hourglass in

Return type

Hourglass object (or null if there are no hourglasses in the model).

Example

To get the first hourglass in model m:

```
var n = Hourglass.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free hourglass label in the model. Also see Hourglass.LastFreeLabel(), Hourglass.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free hourglass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Hourglass label.

Example

To get the first free hourglass label in model m:

```
var label = Hourglass.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the hourglasses in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all hourglasses will be flagged in
flag	Flag	Flag to set on the hourglasses

Return type

No return value

Example

To flag all of the hourglasses with flag f in model m:

```
Hourglass.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the hourglass is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the hourglass

Return type

true if flagged, false if not.

Example

To check if hourglass n has flag f set on it:

```
if (n.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each hourglass in the model.
Note that ForEach has been designed to make looping over hourglasses as fast as possible and so has some limitations.
Firstly, a single temporary Hourglass object is created and on each function call it is updated with the current hourglass data. This means that you should not try to store the Hourglass object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new hourglasses inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all hourglasses are in
func	function	Function to call for each hourglass
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the hourglasses in model m:

```
Hourglass.ForEach(m, test);
function test(n)
{
// n is Hourglass object
}
```

To call function test for all of the hourglasses in model m with optional object:

```
var data = { x:0, y:0 };
Hourglass.ForEach(m, test, data);
function test(n, extra)
{
// n is Hourglass object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Hourglass objects for all of the hourglasses in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get hourglasses from

Return type

Array of Hourglass objects

Example

To make an array of Hourglass objects for all of the hourglasses in model m

```
var n = Hourglass.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a hourglass.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the hourglass n:

```
var comm_array = n.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Hourglass objects for all of the flagged hourglasses in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get hourglasses from
flag	Flag	Flag set on the hourglasses that you want to retrieve

Return type

Array of Hourglass objects

Example

To make an array of Hourglass objects for all of the hourglasses in model m flagged with f

```
var n = Hourglass.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Hourglass object for a hourglass ID.

Arguments

Name	Type	Description
Model	Model	Model to find the hourglass in
number	integer	number of the hourglass you want the Hourglass object for

Return type

Hourglass object (or null if hourglass does not exist).

Example

To get the Hourglass object for hourglass 100 in model m

```
var n = Hourglass.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Hourglass property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Hourglass.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	hourglass property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Hourglass property n.example is a parameter:

```
Options.property_parameter_names = true;
if (n.GetParameter(n.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Hourglass property n.example is a parameter by using the GetParameter method:

```
if (n.ViewParameters().GetParameter(n.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this hourglass (*HOURGLASS). **Note that a carriage return is not added**. See also Hourglass.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for hourglass h:

```
var key = h.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the hourglass. **Note that a carriage return is not added**. See also Hourglass.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for hourglass h:

```
var cards = h.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last hourglass in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last hourglass in

Return type

Hourglass object (or null if there are no hourglasses in the model).

Example

To get the last hourglass in model m:

```
var n = Hourglass.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free hourglass label in the model. Also see Hourglass.FirstFreeLabel(), Hourglass.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free hourglass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Hourglass label.

Example

To get the last free hourglass label in model m:

```
var label = Hourglass.LastFreeLabel(m);
```

Next()

Description

Returns the next hourglass in the model.

Arguments

No arguments

Return type

Hourglass object (or null if there are no more hourglasses in the model).

Example

To get the hourglass in model m after hourglass n:

```
var n = n.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) hourglass label in the model. Also see Hourglass.FirstFreeLabel(), Hourglass.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free hourglass label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Hourglass label.

Example

To get the next free hourglass label in model m:

```
var label = Hourglass.NextFreeLabel(m);
```

Previous()

Description

Returns the previous hourglass in the model.

Arguments

No arguments

Return type

Hourglass object (or null if there are no more hourglasses in the model).

Example

To get the hourglass in model m before hourglass n:

```
var n = n.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the hourglasses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all hourglasses will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the hourglasses in model m, from 1000000:

```
Hourglass.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged hourglasses in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged hourglasses will be renumbered in
flag	Flag	Flag set on the hourglasses that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the hourglasses in model m flagged with f, from 1000000:

```
Hourglass.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select hourglasses using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting hourglasses
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only hourglasses from that model can be selected. If the argument is a Flag then only hourglasses that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any hourglasses can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of hourglasses selected or null if menu cancelled

Example

To select hourglasses from model m, flagging those selected with flag f, giving the prompt 'Select hourglasses':

```
Hourglass.Select(f, 'Select hourglasses', m);
```

To select hourglasses, flagging those selected with flag f but limiting selection to hourglasses flagged with flag l, giving the prompt 'Select hourglasses':

```
Hourglass.Select(f, 'Select hourglasses', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the hourglass.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the hourglass

Return type

No return value

Example

To set flag f for hourglass n:

```
n.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of hourglasses in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing hourglasses should be counted. If false or omitted referenced but undefined hourglasses will also be included in the total.

Return type

number of hourglasses

Example

To get the total number of hourglasses in model m:

```
var total = Hourglass.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the hourglasses in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all hourglasses will be unset in
flag	Flag	Flag to unset on the hourglasses

Return type

No return value

Example

To unset the flag f on all the hourglasses in model m:

```
Hourglass.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Hourglass object.

Example

To check if Hourglass property n.example is a parameter by using the Hourglass.GetParameter() method:

```
if (n.ViewParameters().GetParameter(n.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for hourglass. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for hourglass n:

```
n.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this hourglass.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for hourglass n:

```
var xrefs = n.Xrefs();
```

---

toString()

Description

Creates a string containing the hourglass data in keyword format. Note that this contains the keyword header and the keyword cards. See also Hourglass.Keyword() and Hourglass.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for hourglass h in keyword format

```
var s = h.toString();
```

Include class

The Include class allows you to access the include files in a model. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*, masterInclude (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*, masterInclude (optional)*[boolean]*)
- GetAll(Model*[Model]*, masterInclude (optional)*[boolean]*)
- GetFromID(Model*[Model]*, include number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, Model (optional)*[Model]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- ClearFlag(flag*[Flag]*, clear contents (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetDetailedRange(type argument*[string]*)
- GetLockedLabelData(rangenum*[integer]*)
- IsEmpty()
- Keyword()
- KeywordCards()
- MakeCurrentLayer()
- Modified(listing*[boolean]*)
- Next()
- Previous()
- RemoveLockedLabelData(rangenum*[integer]*)
- SetDetailedRange(type argument*[string]*, min label*[integer]*, max label*[integer]*)
- SetFlag(flag*[Flag]*, flag contents (optional)*[boolean]*)
- SetLockedLabelData(rangenum*[integer]*, min*[integer]*, max*[integer]*, type*[string]*, safe (optional)*[boolean]*, all_includes (optional)*[boolean]*)
- SetTransformOffset(offset*[constant]*, value*[integer]*, check_only (optional)*[boolean]*)
- Warning(message*[string]*, details (optional)*[string]*)
- Write(filename*[string]*, options (optional)*[object]*)
- Write(filename*[string]*, path (optional)*[constant]*, separator (optional)*[constant]*, version (optional)*[string]*, large (optional)*[boolean]*) [deprecated]
- toString()

Include constants

Name	Description
Include.MASTER_ONLY	Only write the master file. See also Model.Write()
Include.MERGE	Merge include files into the master file. See also Model.Write()
Include.NOT_WRITTEN	Prevent include files from being written. See also Model.Write()

Include.SAME_DIR	Write master and include files into the same directory. See also Model.Write()
Include.SELECT	Select include files to be written out. See also Model.Write()
Include.SUBDIR	Write include files to subdirectory. See also Model.Write()

Constants for Directory separators

Name	Description
Include.NATIVE	Use directory separators native to this machine when writing directory names. See also Model.Write()
Include.UNIX	Use unix directory separators when writing directory names. See also Model.Write()
Include.WINDOWS	Use windows directory separators when writing directory names. See also Model.Write()

Constants for Pathnames

Name	Description
Include.ABSOLUTE	Write include file with absolute pathname. See also Model.Write()
Include.RELATIVE	Write include file with relative pathname. See also Model.Write()

Constants for Transformation offsets

Name	Description
Include.ENDOFF	Offset applied to Primer post end keywords (Dummy, Mechanism etc.)
Include.IDDOFF	Offset to define ID (used in Include.SetTransformOffset())
Include.IDEOFF	Offset to element ID (used in Include.SetTransformOffset())
Include.IDFOFF	Offset to function and table ID (used in Include.SetTransformOffset())
Include.IDMOFF	Offset to material ID (used in Include.SetTransformOffset())
Include.IDNOFF	Offset to node ID (used in Include.SetTransformOffset())
Include.IDPOFF	Offset to part ID (used in Include.SetTransformOffset())
Include.IDROFF	Offset to other ID (used in Include.SetTransformOffset())
Include.IDSOFF	Offset to set ID (used in Include.SetTransformOffset())

Constants for compress mode

Name	Description
Include.INDIVIDUAL_GZIP	Each file ’name.key’ is ’gzipped’ to become the individual file ’name.key.gz’
Include.INDIVIDUAL_ZIP	Each file ’name.key’ is ’zipped’ to become the individual file ’name.key.zip’
Include.KEEP_ORIGINAL	Each file ’name.key’ is written using its original compression: uncompressed, ’.gz. or ’.zip’ format

Include properties

Name	Type	Description
comments	string	Comments stored at the top of the include file. Note that this property is not supported for master include file.
fctlen	real	Length transformation factor. Note that this property is not supported for master include file.

fctmas	real	Mass transformation factor. Note that this property is not supported for master include file.
fcttem	string	Temperature transformation factor. Note that this property is not supported for master include file.
fcttim	real	Time transformation factor. Note that this property is not supported for master include file.
file	string	The absolute filename for this include file.Note that this property is not supported for master include file. Also see the name and path properties.
genmax	integer	Include maximum label range value for general items
genmin	integer	Include minimum label range value for general items
iddoff (read only)	integer	Offset to define ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
ideoff (read only)	integer	Offset to element ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idfoff (read only)	integer	Offset to function and table ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idmoff (read only)	integer	Offset to material and equation of state ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idnoff (read only)	integer	Offset to node ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idpoff (read only)	integer	Offset to part, nodal rigid body and constrained node set ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idroff (read only)	integer	Offset to other ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
idsoff (read only)	integer	Offset to set ID. To set property use Include.SetTransformOffset(). Note that this property is not supported for master include file.
incout	integer	Create file containing transformed data. Note that this property is not supported for master include file.
label (read only)	integer	Include number. This number is used to identify the include file. A number is required as it is possible (with include transforms) to have multiple include files with the same name so they cannot be identified by name. The master file is include file number 0. Also see the parent property.
model	integer	The Model number that the include is in.
n_locked_range	integer	Number of locked label ranges. Note that this does not include label ranges locked model-wide (ALL includes).
name	string	The filename for this include file excluding any path. Note that this property is not supported for master include file. Also see the file and path properties.
nelmax	integer	Include maximum label range value for nodes/elements/nrbc/const. spotwelds/define HWA items
nelmin	integer	Include minimum label range value for nodes/elements/nrbc/const. spotwelds/define HWA items
parent	integer	Include number for the parent include file of this include. This number is used to identify the parent include file. A number is required as it is possible (with include transforms) to have multiple include files with the same name so they cannot be identified by name. The master file is include file number 0. Also see the label property. Note that this property is not supported for master include file.
path	string	The path for this include file. Note that this property is not supported for master include file. Also see the file and name properties.
suppressed	logical	If keyout of Include file has been suppressed. Note that this property is not supported for master include file.
tranid	integer	Define transformation number. Note that this property is not supported for master include file.

transform	logical	true if this include file is an *INCLUDE_TRANSFORM, false otherwise. Note that this property is not supported for master include file.

Detailed Description

The Include class allows to create and query include files in a model. See the documentation below for more details.

Constructor

new Include(Model*[Model]*, name*[string]*, parent (optional)*[integer]*)

Description

Create a new Include object.

Arguments

Name	Type	Description
Model	Model	Model that include will be created in
name	string	Include filename
parent (optional)	integer	Parent include file number. If omitted parent will be 0 (main file).

Return type

Include object

Example

To create a new include file /path/to/include.key in model m

```
var i = new Include(m, "/path/to/include.key");
```

Details of functions

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*, masterInclude (optional)*[boolean]*) [static]

Description

Blanks all of the includes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all includes will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().
masterInclude (optional)	boolean	If masterInclude file should be blanked or not. If omitted masterInclude is false. The master file is include file number 0.

Return type

No return value

Example

To blank all of the includes in model m:

```
Include.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged include files in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged includes will be blanked in
flag	Flag	Flag set on the includes that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the include files in model m flagged with f:

```
Include.BlankFlagged(m, f);
```

ClearFlag(flag*[Flag]*, clear contents (optional)*[boolean]*)

Description

Clears a flag on the include.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the include
clear contents (optional)	boolean	If true then the items in the include file will also have flag cleared. If false (default) then the include file contents are not cleared.

Return type

Number of item flags cleared

Example

To clear flag f for include i:

```
i.ClearFlag(f);
```

To clear flag f for include i and all of the items inside the include file, returning the number of item flags cleared in the include file:

```
var ncleared = i.ClearFlag(f, true);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for an include file. For more details on checking see the Check class. Note that this function is not supported for the master include file.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for include i:

```
i.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first include file in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first include in

Return type

Include object (or null if there are no includes in the model).

Example

To get the first include in model m:

```
var i = Include.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*, masterInclude (optional)*[boolean]*) [static]

Description

Flags all of the includes in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all includes will be flagged in
flag	Flag	Flag to set on the includes
masterInclude (optional)	boolean	If masterInclude file should be flagged or not. If omitted masterInclude is false. The master file is include file number 0.

Return type

No return value

Example

To flag all of the includes with flag f in model m:

```
Include.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the include is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the include

Return type

true if flagged, false if not.

Example

To check if include i has flag f set on it:

```
if (i.Flagged(f) ) do_something...
```

GetAll(Model*[Model]*, masterInclude (optional)*[boolean]*) [static]

Description

Returns an array of Include objects for all of the includes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get includes from
masterInclude (optional)	boolean	If masterInclude file should be included or not. If omitted masterInclude is false. The master file is include file number 0.

Return type

Array of Include objects

Example

To make an array of Include objects for all of the includes in model m

```
var i = Include.GetAll(m);
```

GetDetailedRange(type argument*[string]*)

Description

Gets detailed min and max label ranges for specified type from the include.

Arguments

Name	Type	Description
type argument	string	Entity type for which ranges are returned

Return type

An array containing the min and max label ranges for the specified type or null if no range defined for this type.

Example

To get node ranges for include i:

```
var ranges = i.GetDetailedRange("NODE");
var min = ranges[0];
var max = ranges[1];
```

GetFromID(Model*[Model]*, include number*[integer]*) [static]

Description

Returns the Include object for an include label.
Note that items that are in the main keyword file will have a layer value of 0 which can be used as the *include number* argument to this function to return master include file.

Arguments

Name	Type	Description
Model	Model	Model to find the include in
include number	integer	number of the include you want the Include object for

Return type

Include object (or null if include does not exist).

Example

To get the Include object for include 10 in model m

```
var i = Include.GetFromID(m, 10);
```

GetLockedLabelData(rangenum*[integer]*)

Description

Returns the locked label data for include files. Also see the n_locked_range property

Arguments

Name	Type	Description
rangenum	integer	The range number you want the data for; includes can have multiple ranges. **Note that range numbers start at 0, not 1**.

Return type

An array containing the include name (string can also be "ALL" if range is applicable model-wide), start (min) label (integer), end (max) label (integer), safe range (0 or 1 for false or true), and entity type (string).

Example

To get the locked label data for the 3rd range for include i:

```
if (i.locked_range && i.locked_range->n >= 3)
{
    var locked_label_data = i.GetLockedLabelData(2);
}
```

IsEmpty()

Description

Returns true if include is Empty (contains no INSTALLED static/sort/kid/include items).

Arguments

No arguments

Return type

logical

Example

To see if include inc is empty

```
if(inc.Empty())
```

Keyword()

Description

Returns the keyword for this include (*INCLUDE, *INCLUDE_TRANSFORM). **Note that a carriage return is not added**. See also Include.KeywordCards(). This function is not supported for the master include file.

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for include i:

```
var key = i.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the include. **Note that a carriage return is not added**. See also Include.Keyword(). Also note that this function is not supported for the master include file.

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for include i:

```
var cards = i.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last include file in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last include in

Return type

Include object (or null if there are no includes in the model).

Example

To get the last include in model m:

```
var i = Include.Last(m);
```

MakeCurrentLayer()

Description

Sets this include file to be the current layer so that any newly created items are put in this include file. Also see the Model.layer property.

Arguments

No arguments

Return type

No return value

Example

To make include i the current layer:

```
i.MakeCurrentLayer();
```

Modified(listing*[boolean]*)

Description

Returns true if include has been modified.

Arguments

Name	Type	Description
listing	boolean	false for no listing output, true for listing output

Return type

logical

Example

To see if include inc is modified

```
if(inc.Modified(false)) ... (no listing output)
```

Next()

Description

Returns the next include in the model. Note that this function is not supported for the master include file.

Arguments

No arguments

Return type

Include object (or null if there are no more includes in the model).

Example

To get the include in model m after include i:

```
var i = i.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick an include.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only includes from that model can be picked. If the argument is a Flag then only includes that are flagged with *limit* can be selected. If omitted, or null, any includes from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Include object (or null if not picked)

Example

To pick an includee from model m giving the prompt 'Pick include from screen':

```
var i = Include.Pick('Pick include from screen', m);
```

Previous()

Description

Returns the previous include in the model. Note that this function is not supported for the master include file.

Arguments

No arguments

Return type

Include object (or null if there are no more includes in the model).

Example

To get the include in model m before include i:

```
var i = i.Previous();
```

RemoveLockedLabelData(rangenum*[integer]*)

Description

Removes the locked label data for a range in include files. Also see the n_locked_range property

Arguments

Name	Type	Description
rangenum	integer	The locked label range you want to remove. **Note that range numbers start at 0, not 1**.

Return type

No return value.

Example

To remove the locked labels for the 3rd range for include i:

```
i.RemoveLockedLabelData(2);
```

Select(flag*[Flag]*, prompt*[string]*, Model (optional)*[Model]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select includes using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting includes
prompt	string	Text to display as a prompt to the user
Model (optional)	Model	Model to select from
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of items selected or null if menu cancelled

Example

To select an include from model m, flagging those selected with flag f, giving the prompt 'Select include':

```
Include.Select(f, 'Select include', m);
```

SetDetailedRange(type argument*[string]*, min label*[integer]*, max label*[integer]*)

Description

Sets detailed min and max label ranges for specified type on the include.

Arguments

Name	Type	Description
type argument	string	Entity type for which ranges are to be defined
min label	integer	Defines the smallest label for entities of this type
max label	integer	Defines the largest label for entities of this type

Return type

No return value

Example

To set node ranges for include i:

```
i.SetDetailedRange("NODE", 50000, 60000);
```

SetFlag(flag*[Flag]*, flag contents (optional)*[boolean]*)

Description

Sets a flag on the include.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the include
flag contents (optional)	boolean	If true then the items in the include file will also be flagged. If false (default) then the include file contents are not flagged.

Return type

Number of items flagged

Example

To set flag f for include i:

```
i.SetFlag(f);
```

To set flag f for include i and all of the items inside the include file, returning the number of items flagged in the include file:

```
var nflagged = i.SetFlag(f, true);
```

SetLockedLabelData(rangenum*[integer]*, min*[integer]*, max*[integer]*, type*[string]*, safe (optional)*[boolean]*, all_includes (optional)*[boolean]*)

Description

Sets the locked label data for a particular range for an include file. Also see the n_locked_range property

Arguments

Name	Type	Description
rangenum	integer	The range you want to set the data for. **Note that range numbers start at 0, not 1**.
min	integer	Start (min) label for a locked range.
max	integer	End (max) label for a locked range.
type	string	Entity type code - "NODE", "SHELL" etc. Can also be "ALL" (for a list of types see Appendix I of the PRIMER manual).
safe (optional)	boolean	Determines whether a locked range is safe (protected).
all_includes (optional)	boolean	Specified range will be set model-wide (all includes). Only useful when working with the 'master' include.

Return type

No return value.

Example

To set the locked label data for the 3rd range with min 99, max 199, for nodes for include i:

```
i.SetLockedLabelData(3, 99, 199, "NODE");
```

SetTransformOffset(offset*[constant]*, value*[integer]*, check_only (optional)*[boolean]*)

Description

Sets offset values for include transform. This function is required to change the offset values rather than changing the properties directly so that the include can be checked to ensure that the new value does not cause any label clashes with existing items or any negative labels when the transform is unapplied when writing the include. Note that this function is not supported for the master include file.

Arguments

Name	Type	Description
offset	constant	The include transform offset type to change. Can be Include.IDNOFF, Include.IDEOFF, Include.IDPOFF, Include.IDMOFF, Include.IDSOFF, Include.IDFOFF, Include.IDDOFF or Include.IDROFF.
value	integer	The value to change the offset to
check_only (optional)	boolean	Sometimes it may be necessary to check if changing an offset for an include will cause an error or label clash rather than actually changing it. If check only is true then Primer will just check to see if the new value for the offset will cause any label clashes or negative labels **and not change the offset value or any item labels**. If false or omitted then the offset and labels will be updated if there are no errors.

Return type

true if change successful. false if the change would cause a clash of labels or negative labels, in which case the value is not changed.

Example

To set idpoff for include i to 1000, checking that the change is successful:

```
var success = i.SetTransformOffset(Include.IDPOFF, 1000);
```

Total(Model*[Model]*) [static]

Description

Returns the total number of include files in the model.

Arguments

Name	Type	Description
Model	Model	Model to get include total from

Return type

integer

Example

To get the number of include files in model m:

```
var t = Include.Total(m);
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the includes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all includes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the includes in model m:

```
Include.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged include files in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged includes will be unblanked in
flag	Flag	Flag set on the includes that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the include files in model m flagged with f:

```
Include.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the includes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all includes will be unset in
flag	Flag	Flag to unset on the includes

Return type

No return value

Example

To unset the flag f on all of the includes in model m:

```
Include.UnflagAll(m, f);
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for an include file. For more details on checking see the Check class. Note that this function is not supported for the master include file.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add a warning message "My custom warning" for include i:

```
i.Warning("My custom warning");
```

Write(filename*[string]*, options (optional)*[object]*)

Description

Writes an include file. Note that this function is not supported for the master include file.

Arguments

Name	Type	Description

<table>
<tr><td>filename</td><td>string</td><td>Filename of the LS-Dyna keyword file you want to write</td></tr>
<tr><td>options (optional)</td><td>object</td><td>Options specifying how the file should be written out. If omitted the default values below will be used. The properties available are: Object has the following properties:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>binary (optional)</td><td>boolean</td><td>If true then the output file will be written out in binary. If false (default) then an ascii file will be written.</td></tr>
<tr><td>compress (optional)</td><td>boolean</td><td>If true then the output file will be compressed. If false (default) then an uncompressed file will be written.</td></tr>
<tr><td>compressLevel (optional)</td><td>integer</td><td>Compression level for .gz and .zip files. Must be in the range 1 to 9 with 1 being the least compression (fastest speed) to 9 being the greatest compression (slowest speed)</td></tr>
<tr><td>compressMode (optional)</td><td>integer</td><td>This option can be used to specify the mode of compression. Can be Include.KEEP_ORIGINAL or Include.INDIVIDUAL_GZIP or Include.INDIVIDUAL_ZIP</td></tr>
<tr><td>fileStartAscii (optional)</td><td>boolean</td><td>If true then the begining of the file (*CONTROL etc) file is written out in ascii. If false (default) then the entire file is converted to binary.</td></tr>
<tr><td>i10 (optional)</td><td>boolean</td><td>If true then i10 format will be used to write the file. If false (default) then the normal LS-DYNA format will be used.</td></tr>
<tr><td>large (optional)</td><td>boolean</td><td>If true then large format will be used to write the file. If false (default) then the normal LS-DYNA format will be used. Note that large format is only available from version R7.1 and above.</td></tr>
<tr><td>path (optional)</td><td>integer</td><td>The method used to write include paths. Can be Include.ABSOLUTE (default) or Include.RELATIVE</td></tr>
<tr><td>separator (optional)</td><td>integer</td><td>The directory separator used when writing include files. Can be Include.NATIVE (default), Include.UNIX or Include.WINDOWS</td></tr>
<tr><td>version (optional)</td><td>string</td><td>The LS-DYNA version used to write the file. Can be "971R5", "971R4", "971R3", "970v6763" etc (see the version popup in Model->Write '>>> LS-Dyna output options' for a full list). See also Options.dyna_version</td></tr>
</table>
</td></tr>
</table>

Return type

No return value

Example

To Write include i to file /data/test/file.key as a compressed gzip in version R10.0

```
var output_obj = new Object();
output_obj.version = "R10.0";
output_obj.compress = true;
output_obj.compressMode = Include.INDIVIDUAL_GZIP;
i.Write("/data/test/file.key", output_obj);
```

Write(filename*[string]*, path (optional)*[constant]*, separator (optional)*[constant]*, version (optional)*[string]*, large (optional)*[boolean]*) [deprecated]

This function is deprecated in version 15.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Writes an include file. Note that this function is not supported for the master include file.

Arguments

Name	Type	Description
filename	string	Filename of the LS-Dyna keyword file you want to write
path (optional)	constant	The method used to write include paths. Can be Include.ABSOLUTE (default) or Include.RELATIVE
separator (optional)	constant	The directory separator used when writing include files. Can be Include.NATIVE (default), Include.UNIX or Include.WINDOWS
version (optional)	string	The LS-DYNA version used to write the file. Can be "971R5", "971R4", "971R3", "970v6763" etc. (see the version popup in Model->Write '>>> LS-Dyna output options' for a full list). See also Options.dyna_version
large (optional)	boolean	If true then large format will be used to write the file. If false (default) then the normal LS-DYNA format will be used. Note that large format is only available from version R7.1 and above.

Return type

No return value

Example

To write include file i to file /data/test/file.key

```
i.Write("/data/test/file.key");
```

---

toString()

Description

Creates a string containing the include data in keyword format. Note that this contains the keyword header and the keyword cards. See also Include.Keyword() and Include.KeywordCards(). Also note that this function is not supported for the master include file.

Arguments

No arguments

Return type

string

Example

To get data for include i in keyword format

```
var s = i.toString();
```

AxialForceBeam class

The AxialForceBeam class gives you access to initial axial force beam cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

AxialForceBeam properties

Name	Type	Description
bsid	integer	Beam set ID.
exists	logical	true if axial force beam exists, false if referred to but not defined. (read only)
id (read only)	integer	ID of the axial force beam. Only used in Primer.
include	integer	The Include file number that the axial force beam is in.
kbend	integer	Bending stiffness flag.

lcid	integer	Loadcurve ID.
model	integer	The Model number that the axial force beam is in.
scale	real	Scale factor on loadcurve.

Detailed Description

The AxialForceBeam class allows you to create, modify, edit and manipulate initial axial force beam cards. See the documentation below for more details.

Constructor

new AxialForceBeam(Model*[Model]*, bsid*[integer]*, lcid*[integer]*, scale (optional)*[real]*)

Description

Create a new AxialForceBeam object.

Arguments

Name	Type	Description
Model	Model	Model that axial force beam will be created in
bsid	integer	BeamSet ID.
lcid	integer	Loadcurve ID defining preload versus time.
scale (optional)	real	Scale factor on curve

Return type

AxialForceBeam object

Example

To create a new axial force beam in model m using beam set 10, load curve 100:

```
var afb = new AxialForceBeam(m, 10, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a axial force beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the axial force beam

Return type

No return value

Example

To associate comment c to the axial force beam afb:

```
afb.AssociateComment(c);
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the axial force beam.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the axial force beam

Return type

No return value

Example

To clear flag f for axial force beam afb:

```
afb.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the axial force beam.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

AxialForceBeam object

Example

To copy axial force beam afb into axial force beam z:

```
var z = afb.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a axial force beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the axial force beam

Return type

No return value

Example

To detach comment c from the axial force beam afb:

```
afb.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for axial force beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for axial force beam afb:

```
afb.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first axial force beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first axial force beam in

Return type

AxialForceBeam object (or null if there are no axial force beams in the model).

Example

To get the first axial force beam in model m:

```
var afb = AxialForceBeam.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the axial force beams in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all axial force beams will be flagged in

flag	Flag	Flag to set on the axial force beams

Return type

No return value

Example

To flag all of the axial force beams with flag f in model m:

```
AxialForceBeam.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the axial force beam is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the axial force beam

Return type

true if flagged, false if not.

Example

To check if axial force beam afb has flag f set on it:

```
if (afb.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each axial force beam in the model.
Note that ForEach has been designed to make looping over axial force beams as fast as possible and so has some limitations.
Firstly, a single temporary AxialForceBeam object is created and on each function call it is updated with the current axial force beam data. This means that you should not try to store the AxialForceBeam object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new axial force beams inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all axial force beams are in
func	function	Function to call for each axial force beam
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the axial force beams in model m:

```
AxialForceBeam.ForEach(m, test);
function test(afb)
{
// afb is AxialForceBeam object
}
```

To call function test for all of the axial force beams in model m with optional object:

```
var data = { x:0, y:0 };
AxialForceBeam.ForEach(m, test, data);
function test(afb, extra)
{
// afb is AxialForceBeam object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of AxialForceBeam objects for all of the axial force beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get axial force beams from

Return type

Array of AxialForceBeam objects

Example

To make an array of AxialForceBeam objects for all of the axial force beams in model m

```
var afb = AxialForceBeam.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a axial force beam.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the axial force beam afb:

```
var comm_array = afb.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of AxialForceBeam objects for all of the flagged axial force beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get axial force beams from
flag	Flag	Flag set on the axial force beams that you want to retrieve

Return type

Array of AxialForceBeam objects

Example

To make an array of AxialForceBeam objects for all of the axial force beams in model m flagged with f

```
var afb = AxialForceBeam.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the AxialForceBeam object for a axial force beam ID.

Arguments

Name	Type	Description
Model	Model	Model to find the axial force beam in
number	integer	number of the axial force beam you want the AxialForceBeam object for

Return type

AxialForceBeam object (or null if axial force beam does not exist).

Example

To get the AxialForceBeam object for axial force beam 100 in model m

```
var afb = AxialForceBeam.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a AxialForceBeam property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the AxialForceBeam.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	axial force beam property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if AxialForceBeam property afb.example is a parameter:

```
Options.property_parameter_names = true;
if (afb.GetParameter(afb.example) ) do_something...
Options.property_parameter_names = false;
```

To check if AxialForceBeam property afb.example is a parameter by using the GetParameter method:

```
if (afb.ViewParameters().GetParameter(afb.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this axial force beam (*INITIAL_AXIAL_FORCE_BEAM). **Note that a carriage return is not added**. See also AxialForceBeam.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for axial force beam afb:

```
var key = afb.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the axial force beam. **Note that a carriage return is not added**. See also AxialForceBeam.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for axial force beam afb:

```
var cards = afb.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last axial force beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last axial force beam in

Return type

AxialForceBeam object (or null if there are no axial force beams in the model).

Example

To get the last axial force beam in model m:

```
var afb = AxialForceBeam.Last(m);
```

Next()

Description

Returns the next axial force beam in the model.

Arguments

No arguments

Return type

AxialForceBeam object (or null if there are no more axial force beams in the model).

Example

To get the axial force beam in model m after axial force beam afb:

```
var afb = afb.Next();
```

Previous()

Description

Returns the previous axial force beam in the model.

Arguments

No arguments

Return type

AxialForceBeam object (or null if there are no more axial force beams in the model).

Example

To get the axial force beam in model m before axial force beam afb:

```
var afb = afb.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select axial force beams using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting axial force beams
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only axial force beams from that model can be selected. If the argument is a Flag then only axial force beams that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any axial force beams can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of axial force beams selected or null if menu cancelled

Example

To select axial force beams from model m, flagging those selected with flag f, giving the prompt 'Select axial force beams':

```
AxialForceBeam.Select(f, 'Select axial force beams', m);
```

To select axial force beams, flagging those selected with flag f but limiting selection to axial force beams flagged with flag l, giving the prompt 'Select axial force beams':

```
AxialForceBeam.Select(f, 'Select axial force beams', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the axial force beam.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the axial force beam

Return type

No return value

Example

To set flag f for axial force beam afb:

```
afb.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the axial force beam. The axial force beam will be sketched until you either call AxialForceBeam.Unsketch(), AxialForceBeam.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the axial force beam is sketched. If omitted redraw is true. If you want to sketch several axial force beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch axial force beam afb:

```
afb.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged axial force beams in the model. The axial force beams will be sketched until you either call AxialForceBeam.Unsketch(), AxialForceBeam.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged axial force beams will be sketched in
flag	Flag	Flag set on the axial force beams that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the axial force beams are sketched. If omitted redraw is true. If you want to sketch flagged axial force beams several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all axial force beams flagged with flag in model m:

```
AxialForceBeam.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of axial force beams in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing axial force beams should be counted. If false or omitted referenced but undefined axial force beams will also be included in the total.

Return type

number of axial force beams

Example

To get the total number of axial force beams in model m:

```
var total = AxialForceBeam.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the axial force beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all axial force beams will be unset in
flag	Flag	Flag to unset on the axial force beams

Return type

No return value

Example

To unset the flag f on all the axial force beams in model m:

```
AxialForceBeam.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the axial force beam.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the axial force beam is unsketched. If omitted redraw is true. If you want to unsketch several axial force beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch axial force beam afb:

```
afb.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all axial force beams.

Arguments

Name	Type	Description
Model	Model	Model that all axial force beams will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the axial force beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all axial force beams in model m:

```
AxialForceBeam.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged axial force beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all axial force beams will be unsketched in
flag	Flag	Flag set on the axial force beams that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the axial force beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all axial force beams flagged with flag in model m:

```
AxialForceBeam.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

AxialForceBeam object.

Example

To check if AxialForceBeam property afb.example is a parameter by using the AxialForceBeam.GetParameter() method:

```
if (afb.ViewParameters().GetParameter(afb.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for axial force beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for axial force beam afb:

```
afb.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this axial force beam.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for axial force beam afb:

```
var xrefs = afb.Xrefs();
```

---

toString()

Description

Creates a string containing the axial force data in keyword format. Note that this contains the keyword header and the keyword cards. See also AxialForceBeam.Keyword() and AxialForceBeam.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for axial force beam afb in keyword format

```
var s = afb.toString();
```

StressSection class

The StressSection class gives you access to define *INITIAL_STRESS_SECTION cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

StressSection properties

Name	Type	Description
csid	integer	Cross section ID.
exists	logical	true if stress section exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the stress section is in.
issid	integer	StressSection number. Also see the label property which is an alternative name for this.
istiff	integer	Load curve ID defining the artificial stress fraction versus time.
izshear	integer	Shear stress flag.
label	integer	StressSection number. Also see the issid property which is an alternative name for this.
lcid	integer	Load curve ID defining preload stress versus time.
model	integer	The Model number that the stress section is in.
psid	integer	Part set ID.
vid	integer	Vector ID defining the direction normal to the cross section.

Detailed Description

The StressSection class allows you to create, modify, edit and manipulate initial stress section cards. See the documentation below for more details.

Constructor

new StressSection(Model*[Model]*, issid*[integer]*, csid*[integer]*, lcid*[integer]*, psid*[integer]*, vid*[integer]*, izshear*[integer]*, istiff (optional)*[integer]*)

Description

Create a new StressSection object.

Arguments

Name	Type	Description
Model	Model	Model that stress section will be created in
issid	integer	StressSection number.
csid	integer	Cross section ID.
lcid	integer	Load curve ID defining preload stress versus time.
psid	integer	Part set ID.
vid	integer	Vector ID.
izshear	integer	Shear stress flag.
istiff (optional)	integer	Load curve ID defining artificial stress fraction versus time.

Return type

StressSection object

Example

To create a new stress section in model m with label 11, cross section 12, load curve 13, part set 14, vector 15 and shear stress flag 16:

```
var iss = new StressSection(m, 11, 12, 13, 14, 15, 16);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a stress section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the stress section

Return type

No return value

Example

To associate comment c to the stress section iss:

```
iss.AssociateComment(c);
```

Blank()

Description

Blanks the stress section

Arguments

No arguments

Return type

No return value

Example

To blank stress section iss:

```
iss.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the stress sections in model m:

```
StressSection.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged stress sections will be blanked in
flag	Flag	Flag set on the stress sections that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the stress sections in model m flagged with f:

```
StressSection.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the stress section is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if stress section iss is blanked:

```
if (iss.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse stress section iss:

```
iss.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the stress section.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the stress section

Return type

No return value

Example

To clear flag f for stress section iss:

```
iss.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the stress section.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

StressSection object

Example

To copy stress section iss into stress section z:

```
var z = iss.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a initial stress section definition.

Arguments

Name	Type	Description
Model	Model	Model that the stress section will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

StressSection object (or null if not made)

Example

To start creating a initial stress section definition in model m:

```
var iss = StressSection.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a stress section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the stress section

Return type

No return value

Example

To detach comment c from the stress section iss:

```
iss.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit stress section iss:

```
iss.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for stress section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for stress section iss:

```
iss.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first stress section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first stress section in

Return type

StressSection object (or null if there are no stress sections in the model).

Example

To get the first stress section in model m:

```
var iss = StressSection.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free stress section label in the model. Also see StressSection.LastFreeLabel(), StressSection.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free stress section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

StressSection label.

Example

To get the first free stress section label in model m:

```
var label = StressSection.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the stress sections in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be flagged in
flag	Flag	Flag to set on the stress sections

Return type

No return value

Example

To flag all of the stress sections with flag f in model m:

```
StressSection.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the stress section is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the stress section

Return type

true if flagged, false if not.

Example

To check if stress section iss has flag f set on it:

```
if (iss.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each stress section in the model.
Note that ForEach has been designed to make looping over stress sections as fast as possible and so has some limitations.
Firstly, a single temporary StressSection object is created and on each function call it is updated with the current stress section data. This means that you should not try to store the StressSection object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new stress sections inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections are in
func	function	Function to call for each stress section
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the stress sections in model m:

```
StressSection.ForEach(m, test);
function test(iss)
{
// iss is StressSection object
}
```

To call function test for all of the stress sections in model m with optional object:

```
var data = { x:0, y:0 };
StressSection.ForEach(m, test, data);
function test(iss, extra)
{
// iss is StressSection object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of StressSection objects for all of the stress sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get stress sections from

Return type

Array of StressSection objects

Example

To make an array of StressSection objects for all of the stress sections in model m

```
var iss = StressSection.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a stress section.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the stress section iss:

```
var comm_array = iss.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of StressSection objects for all of the flagged stress sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get stress sections from
flag	Flag	Flag set on the stress sections that you want to retrieve

Return type

Array of StressSection objects

Example

To make an array of StressSection objects for all of the stress sections in model m flagged with f

```
var iss = StressSection.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the StressSection object for a stress section ID.

Arguments

Name	Type	Description
Model	Model	Model to find the stress section in
number	integer	number of the stress section you want the StressSection object for

Return type

StressSection object (or null if stress section does not exist).

Example

To get the StressSection object for stress section 100 in model m

```
var iss = StressSection.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a StressSection property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the StressSection.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	stress section property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if StressSection property iss.example is a parameter:

```
Options.property_parameter_names = true;
if (iss.GetParameter(iss.example) ) do_something...
Options.property_parameter_names = false;
```

To check if StressSection property iss.example is a parameter by using the GetParameter method:

```
if (iss.ViewParameters().GetParameter(iss.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this stress section. **Note that a carriage return is not added**. See also StressSection.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for stress section iss:

```
var key = iss.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the stress section. **Note that a carriage return is not added**. See also StressSection.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for stress section iss:

```
var cards = iss.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last stress section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last stress section in

Return type

StressSection object (or null if there are no stress sections in the model).

Example

To get the last stress section in model m:

```
var iss = StressSection.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free stress section label in the model. Also see StressSection.FirstFreeLabel(), StressSection.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free stress section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

StressSection label.

Example

To get the last free stress section label in model m:

```
var label = StressSection.LastFreeLabel(m);
```

Next()

Description

Returns the next stress section in the model.

Arguments

No arguments

Return type

StressSection object (or null if there are no more stress sections in the model).

Example

To get the stress section in model m after stress section iss:

```
var iss = iss.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) stress section label in the model. Also see StressSection.FirstFreeLabel(), StressSection.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free stress section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

StressSection label.

Example

To get the next free stress section label in model m:

```
var label = StressSection.NextFreeLabel(m);
```

Previous()

Description

Returns the previous stress section in the model.

Arguments

No arguments

Return type

StressSection object (or null if there are no more stress sections in the model).

Example

To get the stress section in model m before stress section iss:

```
var iss = iss.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the stress sections in model m, from 1000000:

```
StressSection.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged stress sections will be renumbered in
flag	Flag	Flag set on the stress sections that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the stress sections in model m flagged with f, from 1000000:

```
StressSection.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select stress sections using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting stress sections
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only stress sections from that model can be selected. If the argument is a Flag then only stress sections that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any stress sections can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of stress sections selected or null if menu cancelled

Example

To select stress sections from model m, flagging those selected with flag f, giving the prompt 'Select stress sections':

```
StressSection.Select(f, 'Select stress sections', m);
```

To select stress sections, flagging those selected with flag f but limiting selection to stress sections flagged with flag l, giving the prompt 'Select stress sections':

```
StressSection.Select(f, 'Select stress sections', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the stress section.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the stress section

Return type

No return value

Example

To set flag f for stress section iss:

```
iss.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the stress section. The stress section will be sketched until you either call StressSection.Unsketch(), StressSection.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the stress section is sketched. If omitted redraw is true. If you want to sketch several stress sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch stress section iss:

```
iss.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged stress sections in the model. The stress sections will be sketched until you either call StressSection.Unsketch(), StressSection.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged stress sections will be sketched in
flag	Flag	Flag set on the stress sections that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the stress sections are sketched. If omitted redraw is true. If you want to sketch flagged stress sections several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all stress sections flagged with flag in model m:

```
StressSection.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing stress sections should be counted. If false or omitted referenced but undefined stress sections will also be included in the total.

Return type

number of stress sections

Example

To get the total number of stress sections in model m:

```
var total = StressSection.Total(m);
```

Unblank()

Description

Unblanks the stress section

Arguments

No arguments

Return type

No return value

Example

To unblank stress section iss:

```
iss.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the stress sections in model m:

```
StressSection.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged stress sections will be unblanked in
flag	Flag	Flag set on the stress sections that you want to unblank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the stress sections in model m flagged with f:

```
StressSection.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all stress sections will be unset in
flag	Flag	Flag to unset on the stress sections

Return type

No return value

Example

To unset the flag f on all the stress sections in model m:

```
StressSection.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the stress section.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the stress section is unsketched. If omitted redraw is true. If you want to unsketch several stress sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch stress section iss:

```
iss.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all stress sections.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the stress sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all stress sections in model m:

```
StressSection.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged stress sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all stress sections will be unsketched in
flag	Flag	Flag set on the stress sections that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the stress sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all stress sections flagged with flag in model m:

```
StressSection.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

StressSection object.

Example

To check if StressSection property iss.example is a parameter by using the StressSection.GetParameter() method:

```
if (iss.ViewParameters().GetParameter(iss.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for stress section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for stress section iss:

```
iss.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this stress section.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for stress section iss:

```
var xrefs = iss.Xrefs();
```

---

toString()

Description

Creates a string containing the stress section data in keyword format. Note that this contains the keyword header and the keyword cards. See also StressSection.Keyword() and StressSection.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for stress section iss in keyword format

```
var s = iss.toString();
```

StressShell class

The StressShell class gives you access to define initial stress shell cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetHisvData() [deprecated]
- GetIntegrationPoint(index*[integer]*)
- GetParameter(prop*[string]*)
- GetStressData() [deprecated]
- GetTensrData() [deprecated]
- GetThermalIntegrationPoint(index*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- SetHisvData() [deprecated]
- SetIntegrationPoint(index*[integer]*, data*[Array of data]*)
- SetStressData() [deprecated]
- SetTensrData() [deprecated]
- SetThermalIntegrationPoint(index*[integer]*, data*[Array of data]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

StressShell constants

Name	Description
StressShell.SET	Initial is *INITIAL_STRESS_SHELL_SET.
StressShell.SHELL	Initial is *INITIAL_STRESS_SHELL.

StressShell properties

Name	Type	Description
eid	integer	Shell Element ID or shell set ID
exists	logical	true if stress_shell exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the initial stress shell is in.
large	logical	true if large format, false otherwise
model	integer	The Model number that the initial stress shell is in.
nhisv	integer	Number of additional history variables
nplane	integer	Number of in plane integration points being output
ntensr	integer	Number of components of tensor data taken from the element history variables stored
nthhsv	integer	Number of thermal history variables per thermal integration point
nthick	integer	Number of integration points through the thickness
nthint	integer	Number of thermal integration points
type	constant	The Intial stress shell type. Can be StressShell.SHELL or StressShell.SET.

Detailed Description

The StressShell class allows you to create, modify, edit and manipulate stress_shell cards. See the documentation below for more details.

Constructor

new StressShell(Model*[Model]*, type*[constant]*, eid*[integer]*, nplane*[integer]*, nthick*[integer]*, nhisv*[integer]*, ntensr*[integer]*)

Description

Create a new StressShell object.

Arguments

Name	Type	Description
Model	Model	Model that stress_shell will be created in
type	constant	Specify the type of initial stress shell (Can be StressShell.SHELL or StressShell.SET)
eid	integer	Shell Element ID or shell set ID
nplane	integer	Number of in plane integration points being output
nthick	integer	Number of integration points through the thickness
nhisv	integer	Number of additional history variables
ntensr	integer	Number of components of tensor data taken from the element history variables stored

Return type

StressShell object

Example

To create a new stress_shell in model m, of type SET

```
var s = new StressShell(m, StressShell.SET, 1, 3, 0, 0);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a initial stress shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the initial stress shell

Return type

No return value

Example

To associate comment c to the initial stress shell iss:

```
iss.AssociateComment(c);
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the initial stress shell.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the initial stress shell

Return type

No return value

Example

To clear flag f for initial stress shell iss:

```
iss.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the initial stress shell.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

StressShell object

Example

To copy initial stress shell iss into initial stress shell z:

```
var z = iss.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a initial stress shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the initial stress shell

Return type

No return value

Example

To detach comment c from the initial stress shell iss:

```
iss.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for initial stress shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for initial stress shell iss:

```
iss.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first initial stress shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first initial stress shell in

Return type

StressShell object (or null if there are no initial stress shells in the model).

Example

To get the first initial stress shell in model m:

```
var iss = StressShell.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the initial stress shells in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all initial stress shells will be flagged in
flag	Flag	Flag to set on the initial stress shells

Return type

No return value

Example

To flag all of the initial stress shells with flag f in model m:

```
StressShell.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the initial stress shell is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the initial stress shell

Return type

true if flagged, false if not.

Example

To check if initial stress shell iss has flag f set on it:

```
if (iss.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each initial stress shell in the model.
Note that ForEach has been designed to make looping over initial stress shells as fast as possible and so has some limitations.
Firstly, a single temporary StressShell object is created and on each function call it is updated with the current initial stress shell data. This means that you should not try to store the StressShell object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new initial stress shells inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all initial stress shells are in
func	function	Function to call for each initial stress shell
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the initial stress shells in model m:

```
StressShell.ForEach(m, test);
function test(iss)
{
// iss is StressShell object
}
```

To call function test for all of the initial stress shells in model m with optional object:

```
var data = { x:0, y:0 };
StressShell.ForEach(m, test, data);
function test(iss, extra)
{
// iss is StressShell object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of StressShell objects for all of the initial stress shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial stress shells from

Return type

Array of StressShell objects

Example

To make an array of StressShell objects for all of the initial stress shells in model m

```
var iss = StressShell.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a initial stress shell.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the initial stress shell iss:

```
var comm_array = iss.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of StressShell objects for all of the flagged initial stress shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial stress shells from
flag	Flag	Flag set on the initial stress shells that you want to retrieve

Return type

Array of StressShell objects

Example

To make an array of StressShell objects for all of the initial stress shells in model m flagged with f

```
var iss = StressShell.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the StressShell object for a initial stress shell ID.

Arguments

Name	Type	Description
Model	Model	Model to find the initial stress shell in
number	integer	number of the initial stress shell you want the StressShell object for

Return type

StressShell object (or null if initial stress shell does not exist).

Example

To get the StressShell object for initial stress shell 100 in model m

```
var iss = StressShell.GetFromID(m, 100);
```

GetHisvData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.GetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

GetIntegrationPoint(index*[integer]*)

Description

Returns the data for a specific integration point as an array. For each integration point there will be 8 + nhisv + (6 x ntensr) values. There are nplane x nthick integration points.

Arguments

Name	Type	Description
index	integer	Index you want the integration point data for. **Note that indices start at 0**.

Return type

An array containing the integration point data.

Example

To get the data for the 3rd integration point for initial stress shell iss:

```
var data = iss.GetIntegrationPoint(2);
```

GetParameter(prop*[string]*)

Description

Checks if a StressShell property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the StressShell.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	initial stress shell property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if StressShell property iss.example is a parameter:

```
Options.property_parameter_names = true;
if (iss.GetParameter(iss.example) ) do_something...
Options.property_parameter_names = false;
```

To check if StressShell property iss.example is a parameter by using the GetParameter method:

```
if (iss.ViewParameters().GetParameter(iss.example) ) do_something...
```

GetStressData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.GetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

GetTensrData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.GetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

GetThermalIntegrationPoint(index*[integer]*)

Description

Returns the thermal data for a specific integration point as an array. For each integration point there will be nthhsv values. There are nthint integration points.

Arguments

Name	Type	Description
index	integer	Index you want the integration point data for. **Note that indices start at 0**.

Return type

An array containing the integration point data.

Example

To get the data for the 3rd thermal integration point for initial stress shell iss:

```
var data = iss.GetThermalIntegrationPoint(2);
```

Keyword()

Description

Returns the keyword for this initial stress shell (*INITIAL_STRESS_SHELL). **Note that a carriage return is not added**. See also StressShell.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for stress_shell i:

```
var key = i.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the initial stress shell. **Note that a carriage return is not added**. See also StressShell.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for stress_shell i:

```
var cards = i.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last initial stress shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last initial stress shell in

Return type

StressShell object (or null if there are no initial stress shells in the model).

Example

To get the last initial stress shell in model m:

```
var iss = StressShell.Last(m);
```

Next()

Description

Returns the next initial stress shell in the model.

Arguments

No arguments

Return type

StressShell object (or null if there are no more initial stress shells in the model).

Example

To get the initial stress shell in model m after initial stress shell iss:

```
var iss = iss.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a initial stress shell.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial stress shells from that model can be picked. If the argument is a Flag then only initial stress shells that are flagged with *limit* can be selected. If omitted, or null, any initial stress shells from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

StressShell object (or null if not picked)

Example

To pick a initial stress shell from model m giving the prompt 'Pick initial stress shell from screen':

```
var iss = StressShell.Pick('Pick initial stress shell from screen', m);
```

Previous()

Description

Returns the previous initial stress shell in the model.

Arguments

No arguments

Return type

StressShell object (or null if there are no more initial stress shells in the model).

Example

To get the initial stress shell in model m before initial stress shell iss:

```
var iss = iss.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select initial stress shells using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting initial stress shells
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial stress shells from that model can be selected. If the argument is a Flag then only initial stress shells that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any initial stress shells can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of initial stress shells selected or null if menu cancelled

Example

To select initial stress shells from model m, flagging those selected with flag f, giving the prompt 'Select initial stress shells':

```
StressShell.Select(f, 'Select initial stress shells', m);
```

To select initial stress shells, flagging those selected with flag f but limiting selection to initial stress shells flagged with flag l, giving the prompt 'Select initial stress shells':

```
StressShell.Select(f, 'Select initial stress shells', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the initial stress shell.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the initial stress shell

Return type

No return value

Example

To set flag f for initial stress shell iss:

```
iss.SetFlag(f);
```

SetHisvData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.SetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

SetIntegrationPoint(index*[integer]*, data*[Array of data]*)

Description

Set the data for a specific integration point. For each integration point there will be 8 + nhisv + (6 x ntensr) values. There are nplane x nthick integration points.

Arguments

Name	Type	Description
index	integer	Index you want the integration point data for. **Note that indices start at 0**.
data	Array of data	Array containing the integration point data. The array length should be 8 + nhisv + (6 x ntensr).

Return type

No return value.

Example

To set the 3rd integration point data for initial stress shell iss to the values in array adata:

```
iss.SetIntegrationPoint(2, adata);
```

SetStressData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.SetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

SetTensrData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use StressShell.SetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

SetThermalIntegrationPoint(index*[integer]*, data*[Array of data]*)

Description

Set the thermal data for a specific integration point. For each integration point there will be nthhsv values. There are nthint thermal integration points.

Arguments

Name	Type	Description
index	integer	Index you want the thermal integration point data for. **Note that indices start at 0**.
data	Array of data	Array containing the thermal integration point data. The array length should be nthint.

Return type

No return value.

Example

To set the 3rd thermal integration point data for initial stress shell iss to the values in array adata:

```
iss.SetThermalIntegrationPoint(2, adata);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the initial stress shell. The initial stress shell will be sketched until you either call StressShell.Unsketch(), StressShell.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial stress shell is sketched. If omitted redraw is true. If you want to sketch several initial stress shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch initial stress shell iss:

```
iss.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged initial stress shells in the model. The initial stress shells will be sketched until you either call StressShell.Unsketch(), StressShell.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged initial stress shells will be sketched in
flag	Flag	Flag set on the initial stress shells that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the initial stress shells are sketched. If omitted redraw is true. If you want to sketch flagged initial stress shells several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all initial stress shells flagged with flag in model m:

```
StressShell.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of initial stress shells in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing initial stress shells should be counted. If false or omitted referenced but undefined initial stress shells will also be included in the total.

Return type

number of initial stress shells

Example

To get the total number of initial stress shells in model m:

```
var total = StressShell.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the initial stress shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all initial stress shells will be unset in
flag	Flag	Flag to unset on the initial stress shells

Return type

No return value

Example

To unset the flag f on all the initial stress shells in model m:

```
StressShell.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the initial stress shell.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial stress shell is unsketched. If omitted redraw is true. If you want to unsketch several initial stress shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch initial stress shell iss:

```
iss.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all initial stress shells.

Arguments

Name	Type	Description
Model	Model	Model that all initial stress shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the initial stress shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial stress shells in model m:

```
StressShell.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged initial stress shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial stress shells will be unsketched in
flag	Flag	Flag set on the initial stress shells that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the initial stress shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial stress shells flagged with flag in model m:

```
StressShell.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

StressShell object.

Example

To check if StressShell property iss.example is a parameter by using the StressShell.GetParameter() method:

```
if (iss.ViewParameters().GetParameter(iss.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for initial stress shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for initial stress shell iss:

```
iss.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this initial stress shell.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for initial stress shell iss:

```
var xrefs = iss.Xrefs();
```

---

toString()

Description

Creates a string containing the initial stress shell data in keyword format. Note that this contains the keyword header and the keyword cards. See also StressShell.Keyword() and StressShell.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for stress_shell i in keyword format

```
var s = i.toString();
```

Velocity class

The Velocity class gives you access to define initial velocity cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Velocity properties

Name	Type	Description
boxid	integer	Define box containing nodes

exists	logical	true if velocity exists, false if referred to but not defined. (read only)
icid	integer	Local coordinate system
include	integer	The Include file number that the initial velocity is in.
irigid	integer	IRIGID flag
model	integer	The Model number that the initial velocity is in.
nsid	integer	Set Node set ID
nsidex	integer	Set Exempted Node set ID
vx	real	Initial velocity in X direction
vxe	real	Initial velocity in X direction of exempted nodes
vxr	real	Initial rotational velocity about X axis
vxre	real	Initial rotational velocity about X axis of exempted nodes
vy	real	Initial velocity in Y direction
vye	real	Initial velocity in Y direction of exempted nodes
vyr	real	Initial rotational velocity about Y axis
vyre	real	Initial rotational velocity about Y axis of exempted nodes
vz	real	Initial velocity in Z direction
vze	real	Initial velocity in Z direction of exempted nodes
vzr	real	Initial rotational velocity about Z axis
vzre	real	Initial rotational velocity about Z axis of exempted nodes

Detailed Description

The Velocity class allows you to create, modify, edit and manipulate velocity cards. See the documentation below for more details.

Constructor

new Velocity(Model*[Model]*, nsid*[integer]*, vx*[real]*, vy*[real]*, vz*[real]*, vxr*[real]*, vyr*[real]*, vzr*[real]*, boxid (optional)*[integer]*, irigid (optional)*[integer]*, nsidex (optional)*[integer]*, vxe (optional)*[real]*, vye (optional)*[real]*, vze (optional)*[real]*, vxre (optional)*[real]*, vyre (optional)*[real]*, vzre (optional)*[real]*, icid (optional)*[real]*)

Description

Create a new Velocity object.

Arguments

Name	Type	Description
Model	Model	Model that velocity will be created in
nsid	integer	Set Node set ID
vx	real	Initial velocity in X direction
vy	real	Initial velocity in Y direction
vz	real	Initial velocity in Z direction
vxr	real	Initial rotational velocity about X axis

vyr	real	Initial rotational velocity about Y axis
vzr	real	Initial rotational velocity about Z axis
boxid (optional)	integer	Define box containing nodes
irigid (optional)	integer	IRIGID flag
nsidex (optional)	integer	Set Exempted Node set ID
vxe (optional)	real	Initial velocity in X direction of exempted nodes
vye (optional)	real	Initial velocity in Y direction of exempted nodes
vze (optional)	real	Initial velocity in Z direction of exempted nodes
vxre (optional)	real	Initial rotational velocity about X axis of exempted nodes
vyre (optional)	real	Initial rotational velocity about Y axis of exempted nodes
vzre (optional)	real	Initial rotational velocity about Z axis of exempted nodes
icid (optional)	real	Local coordinate system nodes

Return type

Velocity object

Example

To create a new velocity in model m

```
var s = new Velocity(m, 1, 2.4, 3.7, 7.9, 0.0, 0.0, 0.0);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a initial velocity.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the initial velocity

Return type

No return value

Example

To associate comment c to the initial velocity v:

```
v.AssociateComment(c);
```

Blank()

Description

Blanks the initial velocity

Arguments

No arguments

Return type

No return value

Example

To blank initial velocity v:

```
v.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the initial velocities in model m:

```
Velocity.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged initial velocities will be blanked in
flag	Flag	Flag set on the initial velocities that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the initial velocities in model m flagged with f:

```
Velocity.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the initial velocity is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if initial velocity v is blanked:

```
if (v.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the initial velocity.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the initial velocity

Return type

No return value

Example

To clear flag f for initial velocity v:

```
v.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the initial velocity.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Velocity object

Example

To copy initial velocity v into initial velocity z:

```
var z = v.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a initial velocity.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the initial velocity

Return type

No return value

Example

To detach comment c from the initial velocity v:

```
v.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for initial velocity. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for initial velocity v:

```
v.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first initial velocity in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first initial velocity in

Return type

Velocity object (or null if there are no initial velocities in the model).

Example

To get the first initial velocity in model m:

```
var v = Velocity.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the initial velocities in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities will be flagged in
flag	Flag	Flag to set on the initial velocities

Return type

No return value

Example

To flag all of the initial velocities with flag f in model m:

```
Velocity.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the initial velocity is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the initial velocity

Return type

true if flagged, false if not.

Example

To check if initial velocity v has flag f set on it:

```
if (v.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each initial velocity in the model.
Note that ForEach has been designed to make looping over initial velocities as fast as possible and so has some limitations.
Firstly, a single temporary Velocity object is created and on each function call it is updated with the current initial velocity data. This means that you should not try to store the Velocity object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new initial velocities inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities are in
func	function	Function to call for each initial velocity
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the initial velocities in model m:

```
Velocity.ForEach(m, test);
function test(v)
{
// v is Velocity object
}
```

To call function test for all of the initial velocities in model m with optional object:

```
var data = { x:0, y:0 };
Velocity.ForEach(m, test, data);
function test(v, extra)
{
// v is Velocity object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Velocity objects for all of the initial velocities in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial velocities from

Return type

Array of Velocity objects

Example

To make an array of Velocity objects for all of the initial velocities in model m

```
var v = Velocity.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a initial velocity.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the initial velocity v:

```
var comm_array = v.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Velocity objects for all of the flagged initial velocities in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial velocities from
flag	Flag	Flag set on the initial velocities that you want to retrieve

Return type

Array of Velocity objects

Example

To make an array of Velocity objects for all of the initial velocities in model m flagged with f

```
var v = Velocity.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Velocity object for a initial velocity ID.

Arguments

Name	Type	Description
Model	Model	Model to find the initial velocity in
number	integer	number of the initial velocity you want the Velocity object for

Return type

Velocity object (or null if initial velocity does not exist).

Example

To get the Velocity object for initial velocity 100 in model m

```
var v = Velocity.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Velocity property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Velocity.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	initial velocity property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Velocity property v.example is a parameter:

```
Options.property_parameter_names = true;
if (v.GetParameter(v.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Velocity property v.example is a parameter by using the GetParameter method:

```
if (v.ViewParameters().GetParameter(v.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this initial velocity (*INITIAL_VELOCITY). **Note that a carriage return is not added**. See also Velocity.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for velocity i:

```
var key = i.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the initial velocity. **Note that a carriage return is not added**. See also Velocity.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for velocity i:

```
var cards = i.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last initial velocity in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last initial velocity in

Return type

Velocity object (or null if there are no initial velocities in the model).

Example

To get the last initial velocity in model m:

```
var v = Velocity.Last(m);
```

Next()

Description

Returns the next initial velocity in the model.

Arguments

No arguments

Return type

Velocity object (or null if there are no more initial velocities in the model).

Example

To get the initial velocity in model m after initial velocity v:

```
var v = v.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a initial velocity.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial velocities from that model can be picked. If the argument is a Flag then only initial velocities that are flagged with *limit* can be selected. If omitted, or null, any initial velocities from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Velocity object (or null if not picked)

Example

To pick a initial velocity from model m giving the prompt 'Pick initial velocity from screen':

```
var v = Velocity.Pick('Pick initial velocity from screen', m);
```

Previous()

Description

Returns the previous initial velocity in the model.

Arguments

No arguments

Return type

Velocity object (or null if there are no more initial velocities in the model).

Example

To get the initial velocity in model m before initial velocity v:

```
var v = v.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select initial velocities using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting initial velocities
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial velocities from that model can be selected. If the argument is a Flag then only initial velocities that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any initial velocities can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of initial velocities selected or null if menu cancelled

Example

To select initial velocities from model m, flagging those selected with flag f, giving the prompt 'Select initial velocities':

```
Velocity.Select(f, 'Select initial velocities', m);
```

To select initial velocities, flagging those selected with flag f but limiting selection to initial velocities flagged with flag l, giving the prompt 'Select initial velocities':

```
Velocity.Select(f, 'Select initial velocities', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the initial velocity.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the initial velocity

Return type

No return value

Example

To set flag f for initial velocity v:

```
v.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the initial velocity. The initial velocity will be sketched until you either call Velocity.Unsketch(), Velocity.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity is sketched. If omitted redraw is true. If you want to sketch several initial velocities and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch initial velocity v:

```
v.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged initial velocities in the model. The initial velocities will be sketched until you either call Velocity.Unsketch(), Velocity.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged initial velocities will be sketched in
flag	Flag	Flag set on the initial velocities that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the initial velocities are sketched. If omitted redraw is true. If you want to sketch flagged initial velocities several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all initial velocities flagged with flag in model m:

```
Velocity.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing initial velocities should be counted. If false or omitted referenced but undefined initial velocities will also be included in the total.

Return type

number of initial velocities

Example

To get the total number of initial velocities in model m:

```
var total = Velocity.Total(m);
```

Unblank()

Description

Unblanks the initial velocity

Arguments

No arguments

Return type

No return value

Example

To unblank initial velocity v:

```
v.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the initial velocities in model m:

```
Velocity.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged initial velocities will be unblanked in
flag	Flag	Flag set on the initial velocities that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the initial velocities in model m flagged with f:

```
Velocity.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all initial velocities will be unset in
flag	Flag	Flag to unset on the initial velocities

Return type

No return value

Example

To unset the flag f on all the initial velocities in model m:

```
Velocity.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the initial velocity.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity is unsketched. If omitted redraw is true. If you want to unsketch several initial velocities and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch initial velocity v:

```
v.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all initial velocities.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the initial velocities are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial velocities in model m:

```
Velocity.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged initial velocities in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocities will be unsketched in
flag	Flag	Flag set on the initial velocities that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the initial velocities are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial velocities flagged with flag in model m:

```
Velocity.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Velocity object.

Example

To check if Velocity property v.example is a parameter by using the Velocity.GetParameter() method:

```
if (v.ViewParameters().GetParameter(v.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for initial velocity. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for initial velocity v:

```
v.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this initial velocity.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for initial velocity v:

```
var xrefs = v.Xrefs();
```

---

toString()

Description

Creates a string containing the initial velocity data in keyword format. Note that this contains the keyword header and the keyword cards. See also Velocity.Keyword() and Velocity.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for velocity i in keyword format

```
var s = i.toString();
```

VelocityGeneration class

The VelocityGeneration class gives you access to define initial velocity generation cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

VelocityGeneration constants

Name	Description
VelocityGeneration.NODE_SET	ID is a NODE_SET
VelocityGeneration.PART	ID is a PART
VelocityGeneration.PART_SET	ID is a PART_SET

VelocityGeneration properties

Name	Type	Description
exists	logical	true if velocity exists, false if referred to but not defined. (read only)
icid	integer	Local coordinate system
id	integer	Set Part ID, Part set ID or Node set ID
include	integer	The Include file number that the initial velocity is in.
irigid	integer	Overide part inertia flag
ivatn	integer	Slave parts flag
model	integer	The Model number that the initial velocity generation is in.
nx	real	x-direction cosine
ny	real	y-direction cosine
nz	real	z-direction cosine
omega	real	Angular velocity about the rotational axis
phase	integer	Dynamic relaxation flag
type	constant	Specify the type of Velocity generation (Can be VelocityGeneration.PART_SET or VelocityGeneration.PART or VelocityGeneration.NODE_SET)
vx	real	Initial translational velocity in X direction
vy	real	Initial translational velocity in Y direction
vz	real	Initial translational velocity in Z direction
xc	real	x-coordinate on rotational axis
yc	real	y-coordinate on rotational axis
zc	real	z-coordinate on rotational axis

Detailed Description

The VelocityGeneration class allows you to create, modify, edit and manipulate velocity cards. See the documentation below for more details.

Constructor

new VelocityGeneration(Model*[Model]*, type*[constant]*, id*[integer]*, omega*[real]*, vx*[real]*, vy*[real]*, vz*[real]*, ivatn*[integer]*, xc*[real]*, yc*[real]*, zc*[real]*, nx*[real]*, ny*[real]*, nz*[real]*, phase*[integer]*, irigid*[integer]*, icid*[integer]*)

Description

Create a new VelocityGeneration object.

Arguments

Name	Type	Description
Model	Model	Model that velocity will be created in
type	constant	Specify the type of Velocity generation (Can be VelocityGeneration.PART_SET or VelocityGeneration.PART or VelocityGeneration.NODE_SET)
id	integer	Set Part ID, Part set ID or Node set ID
omega	real	Angular velocity about the rotational axis
vx	real	Initial translational velocity in X direction
vy	real	Initial translational velocity in Y direction
vz	real	Initial translational velocity in Z direction
ivatn	integer	Slave parts flag
xc	real	x-coordinate on rotational axis
yc	real	y-coordinate on rotational axis
zc	real	z-coordinate on rotational axis
nx	real	x-direction cosine
ny	real	y-direction cosine
nz	real	z-direction cosine
phase	integer	Dynamic relaxation flag
irigid	integer	Overide part inertia flag
icid	integer	Local coordinate system

Return type

VelocityGeneration object

Example

To create a new velocity in model m

```
var s = new VelocityGeneration(m, VelocityGeneration.PART, 500, 3.4, 2.4, 3.7,
7.9, 0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1, 0);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a initial velocity generation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the initial velocity generation

Return type

No return value

Example

To associate comment c to the initial velocity generation ivg:

```
ivg.AssociateComment(c);
```

Blank()

Description

Blanks the initial velocity generation

Arguments

No arguments

Return type

No return value

Example

To blank initial velocity generation ivg:

```
ivg.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the initial velocity generations in model m:

```
VelocityGeneration.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged initial velocity generations will be blanked in
flag	Flag	Flag set on the initial velocity generations that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the initial velocity generations in model m flagged with f:

```
VelocityGeneration.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the initial velocity generation is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if initial velocity generation ivg is blanked:

```
if (ivg.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse initial velocity generation ivg:

```
ivg.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the initial velocity generation.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the initial velocity generation

Return type

No return value

Example

To clear flag f for initial velocity generation ivg:

```
ivg.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the initial velocity generation.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

VelocityGeneration object

Example

To copy initial velocity generation ivg into initial velocity generation z:

```
var z = ivg.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an initial velocity generation definition.

Arguments

Name	Type	Description
Model	Model	Model that the initial velocity generation definition will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

VelocityGeneration object (or null if not made)

Example

To start creating an initial velocity generation definition in model m:

```
var v = VelocityGeneration.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a initial velocity generation.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the initial velocity generation

Return type

No return value

Example

To detach comment c from the initial velocity generation ivg:

```
ivg.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit initial velocity generation ivg:

```
ivg.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for initial velocity generation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for initial velocity generation ivg:

```
ivg.Error("My custom error");
```

First(Model[*Model*]) [static]

Description

Returns the first initial velocity generation in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first initial velocity generation in

Return type

VelocityGeneration object (or null if there are no initial velocity generations in the model).

Example

To get the first initial velocity generation in model m:

```
var ivg = VelocityGeneration.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the initial velocity generations in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations will be flagged in
flag	Flag	Flag to set on the initial velocity generations

Return type

No return value

Example

To flag all of the initial velocity generations with flag f in model m:

```
VelocityGeneration.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the initial velocity generation is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the initial velocity generation

Return type

true if flagged, false if not.

Example

To check if initial velocity generation ivg has flag f set on it:

```
if (ivg.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each initial velocity generation in the model.
Note that ForEach has been designed to make looping over initial velocity generations as fast as possible and so has some limitations.
Firstly, a single temporary VelocityGeneration object is created and on each function call it is updated with the current initial velocity generation data. This means that you should not try to store the VelocityGeneration object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new initial velocity generations inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations are in
func	function	Function to call for each initial velocity generation
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the initial velocity generations in model m:

```
VelocityGeneration.ForEach(m, test);
function test(ivg)
{
// ivg is VelocityGeneration object
}
```

To call function test for all of the initial velocity generations in model m with optional object:

```
var data = { x:0, y:0 };
VelocityGeneration.ForEach(m, test, data);
function test(ivg, extra)
{
// ivg is VelocityGeneration object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of VelocityGeneration objects for all of the initial velocity generations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial velocity generations from

Return type

Array of VelocityGeneration objects

Example

To make an array of VelocityGeneration objects for all of the initial velocity generations in model m

```
var ivg = VelocityGeneration.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a initial velocity generation.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the initial velocity generation ivg:

```
var comm_array = ivg.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of VelocityGeneration objects for all of the flagged initial velocity generations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get initial velocity generations from
flag	Flag	Flag set on the initial velocity generations that you want to retrieve

Return type

Array of VelocityGeneration objects

Example

To make an array of VelocityGeneration objects for all of the initial velocity generations in model m flagged with f

```
var ivg = VelocityGeneration.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the VelocityGeneration object for a initial velocity generation ID.

Arguments

Name	Type	Description

Model	Model	Model to find the initial velocity generation in
number	integer	number of the initial velocity generation you want the VelocityGeneration object for

Return type

VelocityGeneration object (or null if initial velocity generation does not exist).

Example

To get the VelocityGeneration object for initial velocity generation 100 in model m

```
var ivg = VelocityGeneration.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a VelocityGeneration property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the VelocityGeneration.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	initial velocity generation property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if VelocityGeneration property ivg.example is a parameter:

```
Options.property_parameter_names = true;
if (ivg.GetParameter(ivg.example) ) do_something...
Options.property_parameter_names = false;
```

To check if VelocityGeneration property ivg.example is a parameter by using the GetParameter method:

```
if (ivg.ViewParameters().GetParameter(ivg.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this initial velocity (*INITIAL_VELOCITY_GENERATION). **Note that a carriage return is not added**. See also VelocityGeneration.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for velocity i:

```
var key = i.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the initial velocity_generation. **Note that a carriage return is not added**. See also VelocityGeneration.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for velocity i:

```
var cards = i.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last initial velocity generation in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last initial velocity generation in

Return type

VelocityGeneration object (or null if there are no initial velocity generations in the model).

Example

To get the last initial velocity generation in model m:

```
var ivg = VelocityGeneration.Last(m);
```

Next()

Description

Returns the next initial velocity generation in the model.

Arguments

No arguments

Return type

VelocityGeneration object (or null if there are no more initial velocity generations in the model).

Example

To get the initial velocity generation in model m after initial velocity generation ivg:

```
var ivg = ivg.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a initial velocity generation.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial velocity generations from that model can be picked. If the argument is a Flag then only initial velocity generations that are flagged with *limit* can be selected. If omitted, or null, any initial velocity generations from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

VelocityGeneration object (or null if not picked)

Example

To pick a initial velocity generation from model m giving the prompt 'Pick initial velocity generation from screen':

```
var ivg = VelocityGeneration.Pick('Pick initial velocity generation from
screen', m);
```

Previous()

Description

Returns the previous initial velocity generation in the model.

Arguments

No arguments

Return type

VelocityGeneration object (or null if there are no more initial velocity generations in the model).

Example

To get the initial velocity generation in model m before initial velocity generation ivg:

```
var ivg = ivg.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select initial velocity generations using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting initial velocity generations
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only initial velocity generations from that model can be selected. If the argument is a Flag then only initial velocity generations that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any initial velocity generations can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of initial velocity generations selected or null if menu cancelled

Example

To select initial velocity generations from model m, flagging those selected with flag f, giving the prompt 'Select initial velocity generations':

```
VelocityGeneration.Select(f, 'Select initial velocity generations', m);
```

To select initial velocity generations, flagging those selected with flag f but limiting selection to initial velocity generations flagged with flag l, giving the prompt 'Select initial velocity generations':

```
VelocityGeneration.Select(f, 'Select initial velocity generations', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the initial velocity generation.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the initial velocity generation

Return type

No return value

Example

To set flag f for initial velocity generation ivg:

```
ivg.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the initial velocity generation. The initial velocity generation will be sketched until you either call VelocityGeneration.Unsketch(), VelocityGeneration.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity generation is sketched. If omitted redraw is true. If you want to sketch several initial velocity generations and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch initial velocity generation ivg:

```
ivg.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged initial velocity generations in the model. The initial velocity generations will be sketched until you either call VelocityGeneration.Unsketch(), VelocityGeneration.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged initial velocity generations will be sketched in
flag	Flag	Flag set on the initial velocity generations that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity generations are sketched. If omitted redraw is true. If you want to sketch flagged initial velocity generations several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all initial velocity generations flagged with flag in model m:

```
VelocityGeneration.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing initial velocity generations should be counted. If false or omitted referenced but undefined initial velocity generations will also be included in the total.

Return type

number of initial velocity generations

Example

To get the total number of initial velocity generations in model m:

```
var total = VelocityGeneration.Total(m);
```

Unblank()

Description

Unblanks the initial velocity generation

Arguments

No arguments

Return type

No return value

Example

To unblank initial velocity generation ivg:

```
ivg.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the initial velocity generations in model m:

```
VelocityGeneration.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged initial velocity generations will be unblanked in
flag	Flag	Flag set on the initial velocity generations that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the initial velocity generations in model m flagged with f:

```
VelocityGeneration.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all initial velocity generations will be unset in
flag	Flag	Flag to unset on the initial velocity generations

Return type

No return value

Example

To unset the flag f on all the initial velocity generations in model m:

```
VelocityGeneration.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the initial velocity generation.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity generation is unsketched. If omitted redraw is true. If you want to unsketch several initial velocity generations and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch initial velocity generation ivg:

```
ivg.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all initial velocity generations.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity generations are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial velocity generations in model m:

```
VelocityGeneration.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged initial velocity generations in the model.

Arguments

Name	Type	Description
Model	Model	Model that all initial velocity generations will be unsketched in
flag	Flag	Flag set on the initial velocity generations that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the initial velocity generations are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all initial velocity generations flagged with flag in model m:

```
VelocityGeneration.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

VelocityGeneration object.

Example

To check if VelocityGeneration property ivg.example is a parameter by using the VelocityGeneration.GetParameter() method:

```
if (ivg.ViewParameters().GetParameter(ivg.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for initial velocity generation. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for initial velocity generation ivg:

```
ivg.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this initial velocity generation.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for initial velocity generation ivg:

```
var xrefs = ivg.Xrefs();
```

---

toString()

Description

Creates a string containing the initial velocity data in keyword format. Note that this contains the keyword header and the keyword cards. See also VelocityGeneration.Keyword() and VelocityGeneration.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for velocity i in keyword format

```
var s = i.toString();
```

IntegrationBeam (IntB) class

The IntegrationBeam class gives you access to integration beam cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetIntegrationPoint(index*[integer]*)
- GetNipCard() [deprecated]
- GetParameter(prop*[string]*)
- GetSectionData() [deprecated]
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- SetIntegrationPoint(index*[integer]*, s*[real]*, t*[real]*, wf*[real]*, pid(optional)*[integer]*)
- SetNipCard() [deprecated]
- SetSectionData() [deprecated]
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

IntegrationBeam properties

Name	Type	Description
d1	real	Cross-section dimension.

d2	real	Cross-section dimension.
d3	real	Cross-section dimension.
d4	real	Cross-section dimension.
d5	real	Cross-section dimension.
d6	real	Cross-section dimension.
exists	logical	true if intb exists, false if referred to but not defined. (read only)
icst	integer	Standard cross section type. If icst is non-zero, nip should be zero and vice-versa.
include	integer	The Include file number that the intb is in.
irid	integer	Integration rule id.
k	integer	Integration refinement parameter for standard cross section types.
model	integer	The Model number that the integration beam is in.
nip	integer	Number of integration points. If nip is non-zero, icst should be zero and vice-versa.
pid	Part	Optional part ID if different from the PID specified on the element card.
ra	real	Relative area of cross section.
s	real	Normalized s coordinate of integration point.
sref	real	Location of reference surface normal to s, for the Hughes-Liu beam only.
t	real	Normalized t coordinate of integration point.
tref	real	Location of reference surface normal to t, for the Hughes-Liu beam only.
wf	real	Weighting factor (area associated with integration point divided by actual cross sectional area).

Detailed Description

The IntegrationBeam class allows you to create, modify, edit and manipulate integration beam cards. See the documentation below for more details.

For convenience "IntB" can also be used as the class name instead of "IntegrationBeam".

Constructor

new IntegrationBeam(Model*[Model]*, irid*[integer]*, nip (optional)*[integer]*, ra (optional)*[real (optional)]*, icst (optional)*[integer]*, k (optional)*[integer]*)

Description

Create a new IntegrationBeam object.

Arguments

Name	Type	Description
Model	Model	Model that intb will be created in.
irid	integer	Integration_Beam ID.
nip (optional)	integer	Number of integration points. If omitted nip will be 0. If nip is non-zero, icst should be zero and vice-versa.
ra (optional)	real (optional)	Relative area of cross section. If omitted ra will be 0.
icst (optional)	integer	Standard cross section type. If omitted icst will be 0. If icst is non-zero, nip should be zero and vice-versa.
k (optional)	integer	Integration refinement parameter for standard cross section types. If omitted k will be 0.

Return type

IntegrationBeam object

Example

To create a new intgb 1000 in model m with the following specifiction: irid, nip, ra, icst, k are 1000, 2, 0.1, 3, 5 respectively

```
var w = new IntegrationBeam(m, 1000, 2, 0.1, 3, 5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a integration beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the integration beam

Return type

No return value

Example

To associate comment c to the integration beam ib:

```
ib.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse integration beam ib:

```
ib.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the integration beam.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the integration beam

Return type

No return value

Example

To clear flag f for integration beam ib:

```
ib.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the integration beam.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

IntegrationBeam object

Example

To copy integration beam ib into integration beam z:

```
var z = ib.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a intb.

Arguments

Name	Type	Description
Model	Model	Model that the intb will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

IntegrationBeam object (or null if not made)

Example

To start creating a intb n in model m:

```
var n = IntegrationBeam.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a integration beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the integration beam

Return type

No return value

Example

To detach comment c from the integration beam ib:

```
ib.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit integration beam ib:

```
ib.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for integration beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for integration beam ib:

```
ib.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first integration beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first integration beam in

Return type

IntegrationBeam object (or null if there are no integration beams in the model).

Example

To get the first integration beam in model m:

```
var ib = IntegrationBeam.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free integration beam label in the model. Also see IntegrationBeam.LastFreeLabel(), IntegrationBeam.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free integration beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

IntegrationBeam label.

Example

To get the first free integration beam label in model m:

```
var label = IntegrationBeam.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the integration beams in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all integration beams will be flagged in
flag	Flag	Flag to set on the integration beams

Return type

No return value

Example

To flag all of the integration beams with flag f in model m:

```
IntegrationBeam.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the integration beam is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the integration beam

Return type

true if flagged, false if not.

Example

To check if integration beam ib has flag f set on it:

```
if (ib.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each integration beam in the model.
Note that ForEach has been designed to make looping over integration beams as fast as possible and so has some limitations.
Firstly, a single temporary IntegrationBeam object is created and on each function call it is updated with the current integration beam data. This means that you should not try to store the IntegrationBeam object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new integration beams inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all integration beams are in
func	function	Function to call for each integration beam
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the integration beams in model m:

```
IntegrationBeam.ForEach(m, test);
function test(ib)
{
// ib is IntegrationBeam object
}
```

To call function test for all of the integration beams in model m with optional object:

```
var data = { x:0, y:0 };
IntegrationBeam.ForEach(m, test, data);
function test(ib, extra)
{
// ib is IntegrationBeam object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of IntegrationBeam objects for all of the integration beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get integration beams from

Return type

Array of IntegrationBeam objects

Example

To make an array of IntegrationBeam objects for all of the integration beams in model m

```
var ib = IntegrationBeam.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a integration beam.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the integration beam ib:

```
var comm_array = ib.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of IntegrationBeam objects for all of the flagged integration beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get integration beams from
flag	Flag	Flag set on the integration beams that you want to retrieve

Return type

Array of IntegrationBeam objects

Example

To make an array of IntegrationBeam objects for all of the integration beams in model m flagged with f

```
var ib = IntegrationBeam.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the IntegrationBeam object for a integration beam ID.

Arguments

Name	Type	Description
Model	Model	Model to find the integration beam in
number	integer	number of the integration beam you want the IntegrationBeam object for

Return type

IntegrationBeam object (or null if integration beam does not exist).

Example

To get the IntegrationBeam object for integration beam 100 in model m

```
var ib = IntegrationBeam.GetFromID(m, 100);
```

GetIntegrationPoint(index*[integer]*)

Description

Returns the data for an integration point in *INTEGRATION_BEAM.**Note data is only available when NIP>0**.

Arguments

Name	Type	Description
index	integer	Index you want the integration point data for. **Note that indices start at 0**.

Return type

An array containing the integration point data.

Example

To get the data for the 3rd integration point for integration beam ib:

```
var data = ib.GetIntegrationPoint(2);
```

GetNipCard() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use IntegrationBeam.GetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

GetParameter(prop*[string]*)

Description

Checks if a IntegrationBeam property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the IntegrationBeam.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	integration beam property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if IntegrationBeam property ib.example is a parameter:

```
Options.property_parameter_names = true;
if (ib.GetParameter(ib.example) ) do_something...
Options.property_parameter_names = false;
```

To check if IntegrationBeam property ib.example is a parameter by using the GetParameter method:

```
if (ib.ViewParameters().GetParameter(ib.example) ) do_something...
```

GetSectionData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use properties d1, d2, sref etc to get the section data.

Arguments

No arguments

Return type

No return value

Keyword()

Description

Returns the keyword for this intb (*INTEGRATION_BEAM). **Note that a carriage return is not added**. See also IntegrationBeam.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for intb n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the intb. **Note that a carriage return is not added**. See also IntegrationBeam.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for intb n:

```
var cards = n.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last integration beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last integration beam in

Return type

IntegrationBeam object (or null if there are no integration beams in the model).

Example

To get the last integration beam in model m:

```
var ib = IntegrationBeam.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free integration beam label in the model. Also see IntegrationBeam.FirstFreeLabel(), IntegrationBeam.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free integration beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

IntegrationBeam label.

Example

To get the last free integration beam label in model m:

```
var label = IntegrationBeam.LastFreeLabel(m);
```

Next()

Description

Returns the next integration beam in the model.

Arguments

No arguments

Return type

IntegrationBeam object (or null if there are no more integration beams in the model).

Example

To get the integration beam in model m after integration beam ib:

```
var ib = ib.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) integration beam label in the model. Also see IntegrationBeam.FirstFreeLabel(), IntegrationBeam.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free integration beam label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

IntegrationBeam label.

Example

To get the next free integration beam label in model m:

```
var label = IntegrationBeam.NextFreeLabel(m);
```

Previous()

Description

Returns the previous integration beam in the model.

Arguments

No arguments

Return type

IntegrationBeam object (or null if there are no more integration beams in the model).

Example

To get the integration beam in model m before integration beam ib:

```
var ib = ib.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the integration beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all integration beams will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the integration beams in model m, from 1000000:

```
IntegrationBeam.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged integration beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged integration beams will be renumbered in
flag	Flag	Flag set on the integration beams that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the integration beams in model m flagged with f, from 1000000:

```
IntegrationBeam.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select integration beams using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting integration beams
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only integration beams from that model can be selected. If the argument is a Flag then only integration beams that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any integration beams can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of integration beams selected or null if menu cancelled

Example

To select integration beams from model m, flagging those selected with flag f, giving the prompt 'Select integration beams':

```
IntegrationBeam.Select(f, 'Select integration beams', m);
```

To select integration beams, flagging those selected with flag f but limiting selection to integration beams flagged with flag l, giving the prompt 'Select integration beams':

```
IntegrationBeam.Select(f, 'Select integration beams', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the integration beam.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the integration beam

Return type

No return value

Example

To set flag f for integration beam ib:

```
ib.SetFlag(f);
```

SetIntegrationPoint(index*[integer]*, s*[real]*, t*[real]*, wf*[real]*, pid(optional)*[integer]*)

Description

Sets the integration point data for an *INTEGRATION_BEAM.

Arguments

Name	Type	Description
index	integer	Index you want to set the integration point data for. **Note that indices start at 0**.
s	real	s coordinate of integration point in range -1 to 1.
t	real	s coordinate of integration point in range -1 to 1.
wf	real	Weighting factor, area associated with the integration point divided by actual beam cross sectional area.
pid(optional)	integer	Optional part ID if different from the PID specified on the element card.

Return type

No return value.

Example

To set the 4th integration point for *INTEGRATION_BEAM ib to the following specifiction: s, t, wf, pid are 0.1, 0.2, 0.3, 1 respectively

```
ib.SetIntegrationPoint(3, 0.1, 0.2, 0.3, 1);
```

SetNipCard() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use IntegrationBeam.SetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

SetSectionData() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use properties d1, d2, sref etc to set the section data.

Arguments

No arguments

Return type

No return value

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of integration beams in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing integration beams should be counted. If false or omitted referenced but undefined integration beams will also be included in the total.

Return type

number of integration beams

Example

To get the total number of integration beams in model m:

```
var total = IntegrationBeam.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the integration beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all integration beams will be unset in
flag	Flag	Flag to unset on the integration beams

Return type

No return value

Example

To unset the flag f on all the integration beams in model m:

```
IntegrationBeam.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

IntegrationBeam object.

Example

To check if IntegrationBeam property ib.example is a parameter by using the IntegrationBeam.GetParameter() method:

```
if (ib.ViewParameters().GetParameter(ib.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for integration beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for integration beam ib:

```
ib.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this integration beam.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for integration beam ib:

```
var xrefs = ib.Xrefs();
```

---

toString()

Description

Creates a string containing the intb data in keyword format. Note that this contains the keyword header and the keyword cards. See also IntegrationBeam.Keyword() and IntegrationBeam.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for intb n in keyword format

```
var s = n.toString();
```

IntegrationShell (IntS) class

The IntegrationShell class gives you access to integration shell cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetIntegrationPoint(index*[integer]*)
- GetNipCard() [deprecated]
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- SetIntegrationPoint(index*[integer]*, s*[real]*, wf*[real]*, pid(optional)*[integer]*)
- SetNipCard() [deprecated]
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

IntegrationShell properties

Name	Type	Description
esop	integer	Equal spacing of integration points option.
exists	logical	true if ints exists, false if referred to but not defined. (read only)

failopt	integer	Treatment of failure when mixing different constitutive types.
include	integer	The Include file number that the ints is in.
irid	integer	Integration rule id.
model	integer	The Model number that the integration shell is in.
nip	integer	Number of integration points.
pid	Part	Optional part ID if different from the PID specified on the element card.
s	real	Coordinate of integration point in range -1 to 1.
wf	real	Weighting factor (thickness associated with integration point divided by actual shell thickness).

Detailed Description

The IntegrationShell class allows you to create, modify, edit and manipulate integration shell cards. See the documentation below for more details.

For convenience "IntS" can also be used as the class name instead of "IntegrationShell".

Constructor

new IntegrationShell(Model*[Model]*, irid*[integer]*, nip*[integer]*, esop (optional)*[integer]*, failopt (optional)*[integer]*)

Description

Create a new IntegrationShell object.

Arguments

Name	Type	Description
Model	Model	Model that ints will be created in.
irid	integer	Integration_Beam ID.
nip	integer	Number of integration points.
esop (optional)	integer	Equal spacing of integration points option. If omitted esop will be 0.
failopt (optional)	integer	Treatment of failure when mixing different constitutive types. If omitted failopt will be 0.

Return type

IntegrationShell object

Example

To create a new ints 1000 in model m with the following specifiction: irid, nip, esop, failopt are 1000, 2, 0, 1 respectively

```
var w = new IntegrationBeam(m, 1000, 2, 0, 1);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a integration shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the integration shell

Return type

No return value

Example

To associate comment c to the integration shell is:

```
is.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse integration shell is:

```
is.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the integration shell.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the integration shell

Return type

No return value

Example

To clear flag f for integration shell is:

```
is.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the integration shell.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

IntegrationShell object

Example

To copy integration shell is into integration shell z:

```
var z = is.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a ints.

Arguments

Name	Type	Description
Model	Model	Model that the ints will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

IntegrationShell object (or null if not made)

Example

To start creating an integration shell is in model m:

```
var is = IntegrationShell.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a integration shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the integration shell

Return type

No return value

Example

To detach comment c from the integration shell is:

```
is.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit integration shell is:

```
is.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for integration shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for integration shell is:

```
is.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first integration shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first integration shell in

Return type

IntegrationShell object (or null if there are no integration shells in the model).

Example

To get the first integration shell in model m:

```
var is = IntegrationShell.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free integration shell label in the model. Also see IntegrationShell.LastFreeLabel(), IntegrationShell.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free integration shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

IntegrationShell label.

Example

To get the first free integration shell label in model m:

```
var label = IntegrationShell.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the integration shells in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all integration shells will be flagged in
flag	Flag	Flag to set on the integration shells

Return type

No return value

Example

To flag all of the integration shells with flag f in model m:

```
IntegrationShell.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the integration shell is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the integration shell

Return type

true if flagged, false if not.

Example

To check if integration shell is has flag f set on it:

```
if (is.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each integration shell in the model.
Note that ForEach has been designed to make looping over integration shells as fast as possible and so has some limitations.
Firstly, a single temporary IntegrationShell object is created and on each function call it is updated with the current integration shell data. This means that you should not try to store the IntegrationShell object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new integration shells inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all integration shells are in
func	function	Function to call for each integration shell
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the integration shells in model m:

```
IntegrationShell.ForEach(m, test);
function test(is)
{
// is is IntegrationShell object
}
```

To call function test for all of the integration shells in model m with optional object:

```
var data = { x:0, y:0 };
IntegrationShell.ForEach(m, test, data);
function test(is, extra)
{
// is is IntegrationShell object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of IntegrationShell objects for all of the integration shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get integration shells from

Return type

Array of IntegrationShell objects

Example

To make an array of IntegrationShell objects for all of the integration shells in model m

```
var is = IntegrationShell.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a integration shell.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the integration shell is:

```
var comm_array = is.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of IntegrationShell objects for all of the flagged integration shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get integration shells from
flag	Flag	Flag set on the integration shells that you want to retrieve

Return type

Array of IntegrationShell objects

Example

To make an array of IntegrationShell objects for all of the integration shells in model m flagged with f

```
var is = IntegrationShell.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the IntegrationShell object for a integration shell ID.

Arguments

Name	Type	Description
Model	Model	Model to find the integration shell in
number	integer	number of the integration shell you want the IntegrationShell object for

Return type

IntegrationShell object (or null if integration shell does not exist).

Example

To get the IntegrationShell object for integration shell 100 in model m

```
var is = IntegrationShell.GetFromID(m, 100);
```

GetIntegrationPoint(index*[integer]*)

Description

Returns the data for an integration point in *INTEGRATION_SHELL.**Note data is only available when NIP>0 and ESOP=0**.

Arguments

Name	Type	Description
index	integer	Index you want the integration point data for. **Note that indices start at 0**.

Return type

An array containing the integration point data.

Example

To get the data for the 3rd integration point for integration shell:

```
var data = is.GetIntegrationPoint(2);
```

GetNipCard() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use IntegrationShell.GetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

GetParameter(prop*[string]*)

Description

Checks if a IntegrationShell property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the IntegrationShell.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	integration shell property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if IntegrationShell property is.example is a parameter:

```
Options.property_parameter_names = true;
if (is.GetParameter(is.example) ) do_something...
Options.property_parameter_names = false;
```

To check if IntegrationShell property is.example is a parameter by using the GetParameter method:

```
if (is.ViewParameters().GetParameter(is.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this ints (*INTEGRATION_SHELL). **Note that a carriage return is not added**. See also IntegrationShell.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for ints n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the ints. **Note that a carriage return is not added**. See also IntegrationShell.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for integration shell is:

```
var cards = is.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last integration shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last integration shell in

Return type

IntegrationShell object (or null if there are no integration shells in the model).

Example

To get the last integration shell in model m:

```
var is = IntegrationShell.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free integration shell label in the model. Also see IntegrationShell.FirstFreeLabel(), IntegrationShell.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free integration shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

IntegrationShell label.

Example

To get the last free integration shell label in model m:

```
var label = IntegrationShell.LastFreeLabel(m);
```

Next()

Description

Returns the next integration shell in the model.

Arguments

No arguments

Return type

IntegrationShell object (or null if there are no more integration shells in the model).

Example

To get the integration shell in model m after integration shell is:

```
var is = is.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) integration shell label in the model. Also see IntegrationShell.FirstFreeLabel(), IntegrationShell.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free integration shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

IntegrationShell label.

Example

To get the next free integration shell label in model m:

```
var label = IntegrationShell.NextFreeLabel(m);
```

Previous()

Description

Returns the previous integration shell in the model.

Arguments

No arguments

Return type

IntegrationShell object (or null if there are no more integration shells in the model).

Example

To get the integration shell in model m before integration shell is:

```
var is = is.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the integration shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all integration shells will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the integration shells in model m, from 1000000:

```
IntegrationShell.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged integration shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged integration shells will be renumbered in
flag	Flag	Flag set on the integration shells that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the integration shells in model m flagged with f, from 1000000:

```
IntegrationShell.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select integration shells using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting integration shells
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only integration shells from that model can be selected. If the argument is a Flag then only integration shells that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any integration shells can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of integration shells selected or null if menu cancelled

Example

To select integration shells from model m, flagging those selected with flag f, giving the prompt 'Select integration shells':

```
IntegrationShell.Select(f, 'Select integration shells', m);
```

To select integration shells, flagging those selected with flag f but limiting selection to integration shells flagged with flag l, giving the prompt 'Select integration shells':

```
IntegrationShell.Select(f, 'Select integration shells', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the integration shell.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the integration shell

Return type

No return value

Example

To set flag f for integration shell is:

```
is.SetFlag(f);
```

SetIntegrationPoint(index*[integer]*, s*[real]*, wf*[real]*, pid(optional)*[integer]*)

Description

Sets the integration point data for an *INTEGRATION_SHELL.

Arguments

Name	Type	Description
index	integer	Index you want to set the integration point data for. **Note that indices start at 0**.
s	real	Coordinate of integration point in range -1 to 1.
wf	real	Weighting factor, thickness associated with the integration point divided by actual shell thickness.
pid(optional)	integer	Optional part ID if different from the PID specified on the element card.

Return type

No return value.

Example

To set the 4th integration point for *INTEGRATION_SHELL is to the following specifiction: s, wf, pid are 0.1, 0.2, 1 respectively

```
is.SetIntegrationPoint(3, 0.1, 0.2, 1);
```

SetNipCard() [deprecated]

This function is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Please use IntegrationShell.SetIntegrationPoint() instead.

Arguments

No arguments

Return type

No return value

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of integration shells in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing integration shells should be counted. If false or omitted referenced but undefined integration shells will also be included in the total.

Return type

number of integration shells

Example

To get the total number of integration shells in model m:

```
var total = IntegrationShell.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the integration shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all integration shells will be unset in
flag	Flag	Flag to unset on the integration shells

Return type

No return value

Example

To unset the flag f on all the integration shells in model m:

```
IntegrationShell.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

IntegrationShell object.

Example

To check if IntegrationShell property is.example is a parameter by using the IntegrationShell.GetParameter() method:

```
if (is.ViewParameters().GetParameter(is.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for integration shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for integration shell is:

```
is.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this integration shell.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for integration shell is:

```
var xrefs = is.Xrefs();
```

---

toString()

Description

Creates a string containing the ints data in keyword format. Note that this contains the keyword header and the keyword cards. See also IntegrationShell.Keyword() and IntegrationShell.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for integration shell is in keyword format

```
var s = is.toString();
```

InterfaceComponent class

The InterfaceComponent class gives you access to interface component cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

InterfaceComponent constants

Name	Description
InterfaceComponent.NODE	Node option
InterfaceComponent.SEGMENT	Segment option

InterfaceComponent properties

Name	Type	Description
cid	integer	Coordinate system ID.
exists	logical	true if interface component exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the interface component is in.
model	integer	The Model number that the interface component is in.
nid	integer	Node ID.
nsid	integer	Element ID or element set ID. The ssid property is an alternative name for this.
option	constant	InterfaceComponent option. Can be InterfaceComponent.NODE, InterfaceComponent.SEGMENT,
ssid	integer	Element ID or element set ID. The nsid property is an alternative name for this.
title	string	InterfaceComponent title

Detailed Description

The InterfaceComponent class allows you to create, modify, edit and manipulate interface component cards. See the documentation below for more details.

Constructor

new InterfaceComponent(Model*[Model]*, type*[constant]*, snid/ssid*[integer]*, cid*[integer]*, nid*[integer]*, label (optional)*[integer]*, title (optional)*[string]*)

Description

Create a new InterfaceComponent object.

Arguments

Name	Type	Description
Model	Model	Model that InterfaceComponent will be created in
type	constant	InterfaceComponent type. Can be InterfaceComponent.NODE, InterfaceComponent.SEGMENT,
snid/ssid	integer	Set node or set segment ID
cid	integer	Coordinate system ID
nid	integer	Node ID
label (optional)	integer	InterfaceComponent number
title (optional)	string	Title for this interface

Return type

InterfaceComponent object

Example

To create a new Interface Component in model m with option: NODE, nsid: 100, cyd: 200, nid: 300, ID: 1, title: "MyInterfaceComponent"

```
var i_c = new InterfaceComponent(m, InterfaceComponent.NODE, 100, 200, 300, 1,
MyInterfaceComponent);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a interface component.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the interface component

Return type

No return value

Example

To associate comment c to the interface component i_c:

```
i_c.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse interface component i_c:

```
i_c.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the interface component.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the interface component

Return type

No return value

Example

To clear flag f for interface component i_c:

```
i_c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the interface component.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

InterfaceComponent object

Example

To copy interface component i_c into interface component z:

```
var z = i_c.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an InterfaceComponent.

Arguments

Name	Type	Description
Model	Model	Model that the InterfaceComponent will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

InterfaceComponent object (or null if not made)

Example

To start creating an InterfaceComponent in model m:

```
var ed = InterfaceComponent.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a interface component.

Arguments

Name	Type	Description

Comment	Comment	Comment that will be detached from the interface component

Return type

No return value

Example

To detach comment c from the interface component i_c:

```
i_c.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit interface component i_c:

```
i_c.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for interface component. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for interface component i_c:

```
i_c.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first interface component in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first interface component in

Return type

InterfaceComponent object (or null if there are no interface components in the model).

Example

To get the first interface component in model m:

```
var i_c = InterfaceComponent.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free interface component label in the model. Also see InterfaceComponent.LastFreeLabel(), InterfaceComponent.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free interface component label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

InterfaceComponent label.

Example

To get the first free interface component label in model m:

```
var label = InterfaceComponent.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the interface components in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all interface components will be flagged in
flag	Flag	Flag to set on the interface components

Return type

No return value

Example

To flag all of the interface components with flag f in model m:

```
InterfaceComponent.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the interface component is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the interface component

Return type

true if flagged, false if not.

Example

To check if interface component i_c has flag f set on it:

```
if (i_c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each interface component in the model.
Note that ForEach has been designed to make looping over interface components as fast as possible and so has some limitations.
Firstly, a single temporary InterfaceComponent object is created and on each function call it is updated with the current interface component data. This means that you should not try to store the InterfaceComponent object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new interface components inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all interface components are in
func	function	Function to call for each interface component
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the interface components in model m:

```
InterfaceComponent.ForEach(m, test);
function test(i_c)
{
// i_c is InterfaceComponent object
}
```

To call function test for all of the interface components in model m with optional object:

```
var data = { x:0, y:0 };
InterfaceComponent.ForEach(m, test, data);
function test(i_c, extra)
{
// i_c is InterfaceComponent object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of InterfaceComponent objects for all of the interface components in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interface components from

Return type

Array of InterfaceComponent objects

Example

To make an array of InterfaceComponent objects for all of the interface components in model m

```
var i_c = InterfaceComponent.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a interface component.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the interface component i_c:

```
var comm_array = i_c.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of InterfaceComponent objects for all of the flagged interface components in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interface components from
flag	Flag	Flag set on the interface components that you want to retrieve

Return type

Array of InterfaceComponent objects

Example

To make an array of InterfaceComponent objects for all of the interface components in model m flagged with f

```
var i_c = InterfaceComponent.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the InterfaceComponent object for a interface component ID.

Arguments

Name	Type	Description
Model	Model	Model to find the interface component in
number	integer	number of the interface component you want the InterfaceComponent object for

Return type

InterfaceComponent object (or null if interface component does not exist).

Example

To get the InterfaceComponent object for interface component 100 in model m

```
var i_c = InterfaceComponent.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a InterfaceComponent property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the InterfaceComponent.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	interface component property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if InterfaceComponent property i_c.example is a parameter:

```
Options.property_parameter_names = true;
if (i_c.GetParameter(i_c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if InterfaceComponent property i_c.example is a parameter by using the GetParameter method:

```
if (i_c.ViewParameters().GetParameter(i_c.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this InterfaceComponent (*INTERFACE_COMPONENT). **Note that a carriage return is not added**. See also InterfaceComponent.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for InterfaceComponent ed:

```
var key = ed.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the InterfaceComponent. **Note that a carriage return is not added**. See also InterfaceComponent.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for InterfaceComponent ed:

```
var cards = ed.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last interface component in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last interface component in

Return type

InterfaceComponent object (or null if there are no interface components in the model).

Example

To get the last interface component in model m:

```
var i_c = InterfaceComponent.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free interface component label in the model. Also see InterfaceComponent.FirstFreeLabel(), InterfaceComponent.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free interface component label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

InterfaceComponent label.

Example

To get the last free interface component label in model m:

```
var label = InterfaceComponent.LastFreeLabel(m);
```

Next()

Description

Returns the next interface component in the model.

Arguments

No arguments

Return type

InterfaceComponent object (or null if there are no more interface components in the model).

Example

To get the interface component in model m after interface component i_c:

```
var i_c = i_c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) interface component label in the model. Also see InterfaceComponent.FirstFreeLabel(), InterfaceComponent.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free interface component label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

InterfaceComponent label.

Example

To get the next free interface component label in model m:

```
var label = InterfaceComponent.NextFreeLabel(m);
```

Previous()

Description

Returns the previous interface component in the model.

Arguments

No arguments

Return type

InterfaceComponent object (or null if there are no more interface components in the model).

Example

To get the interface component in model m before interface component i_c:

```
var i_c = i_c.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the interface components in the model.

Arguments

Name	Type	Description
Model	Model	Model that all interface components will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the interface components in model m, from 1000000:

```
InterfaceComponent.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged interface components in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged interface components will be renumbered in
flag	Flag	Flag set on the interface components that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the interface components in model m flagged with f, from 1000000:

```
InterfaceComponent.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select interface components using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting interface components
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only interface components from that model can be selected. If the argument is a Flag then only interface components that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any interface components can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of interface components selected or null if menu cancelled

Example

To select interface components from model m, flagging those selected with flag f, giving the prompt 'Select interface components':

```
InterfaceComponent.Select(f, 'Select interface components', m);
```

To select interface components, flagging those selected with flag f but limiting selection to interface components flagged with flag l, giving the prompt 'Select interface components':

```
InterfaceComponent.Select(f, 'Select interface components', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the interface component.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the interface component

Return type

No return value

Example

To set flag f for interface component i_c:

```
i_c.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of interface components in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing interface components should be counted. If false or omitted referenced but undefined interface components will also be included in the total.

Return type

number of interface components

Example

To get the total number of interface components in model m:

```
var total = InterfaceComponent.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the interface components in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all interface components will be unset in
flag	Flag	Flag to unset on the interface components

Return type

No return value

Example

To unset the flag f on all the interface components in model m:

```
InterfaceComponent.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

InterfaceComponent object.

Example

To check if InterfaceComponent property i_c.example is a parameter by using the InterfaceComponent.GetParameter() method:

```
if (i_c.ViewParameters().GetParameter(i_c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for interface component. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for interface component i_c:

```
i_c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this interface component.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for interface component i_c:

```
var xrefs = i_c.Xrefs();
```

---

toString()

Description

Creates a string containing the InterfaceComponent data in keyword format. Note that this contains the keyword header and the keyword cards. See also InterfaceComponent.Keyword() and InterfaceComponent.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for InterfaceComponent ed in keyword format

```
var s = ed.toString();
```

InterfaceLinkingEdge class

The InterfaceLinkingEdge class gives you access to define Interface Linking Edge cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

InterfaceLinkingEdge properties

Name	Type	Description
exists	logical	true if Interface Linking Edge exists, false if referred to but not defined. (read only)
ifid	integer	Interface ID.
include	integer	The Include file number that the Interface Linking Edge is in.
model	integer	The Model number that the Interface Linking Edge is in.
nsid	integer	Node set ID

Detailed Description

The InterfaceLinkingEdge class allows you to create, modify, edit and manipulate Interface Linking Edge cards. See the documentation below for more details.

Constructor

new InterfaceLinkingEdge(Model*[Model]*, nsid*[integer]*, ifid*[integer]*)

Description

Create a new InterfaceLinkingEdge object.

Arguments

Name	Type	Description
Model	Model	Model that Interface Linking Edge will be created in
nsid	integer	Node set ID
ifid	integer	Interface ID

Return type

InterfaceLinkingEdge object

Example

To create a new Interface Linking Edge in model m with NSID 900 and IFID 2

```
var b = new InterfaceLinkingEdge(m, 900, 2);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a Interface Linking Edge.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the Interface Linking Edge

Return type

No return value

Example

To associate comment c to the Interface Linking Edge I_LE:

```
I_LE.AssociateComment(c);
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the Interface Linking Edge.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the Interface Linking Edge

Return type

No return value

Example

To clear flag f for Interface Linking Edge I_LE:

```
I_LE.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the Interface Linking Edge.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

InterfaceLinkingEdge object

Example

To copy Interface Linking Edge I_LE into Interface Linking Edge z:

```
var z = I_LE.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a Interface Linking Edge.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the Interface Linking Edge

Return type

No return value

Example

To detach comment c from the Interface Linking Edge I_LE:

```
I_LE.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for Interface Linking Edge. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for Interface Linking Edge I_LE:

```
I_LE.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first Interface Linking Edge in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first Interface Linking Edge in

Return type

InterfaceLinkingEdge object (or null if there are no Interface Linking Edges in the model).

Example

To get the first Interface Linking Edge in model m:

```
var I_LE = InterfaceLinkingEdge.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the Interface Linking Edges in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all Interface Linking Edges will be flagged in
flag	Flag	Flag to set on the Interface Linking Edges

Return type

No return value

Example

To flag all of the Interface Linking Edges with flag f in model m:

```
InterfaceLinkingEdge.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the Interface Linking Edge is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the Interface Linking Edge

Return type

true if flagged, false if not.

Example

To check if Interface Linking Edge I_LE has flag f set on it:

```
if (I_LE.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each Interface Linking Edge in the model.
Note that ForEach has been designed to make looping over Interface Linking Edges as fast as possible and so has some limitations.
Firstly, a single temporary InterfaceLinkingEdge object is created and on each function call it is updated with the current Interface Linking Edge data. This means that you should not try to store the InterfaceLinkingEdge object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new Interface Linking Edges inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all Interface Linking Edges are in
func	function	Function to call for each Interface Linking Edge
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the Interface Linking Edges in model m:

```
InterfaceLinkingEdge.ForEach(m, test);
function test(I_LE)
{
// I_LE is InterfaceLinkingEdge object
}
```

To call function test for all of the Interface Linking Edges in model m with optional object:

```
var data = { x:0, y:0 };
InterfaceLinkingEdge.ForEach(m, test, data);
function test(I_LE, extra)
{
// I_LE is InterfaceLinkingEdge object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of InterfaceLinkingEdge objects for all of the Interface Linking Edges in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get Interface Linking Edges from

Return type

Array of InterfaceLinkingEdge objects

Example

To make an array of InterfaceLinkingEdge objects for all of the Interface Linking Edges in model m

```
var I_LE = InterfaceLinkingEdge.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a Interface Linking Edge.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the Interface Linking Edge I_LE:

```
var comm_array = I_LE.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of InterfaceLinkingEdge objects for all of the flagged Interface Linking Edges in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get Interface Linking Edges from
flag	Flag	Flag set on the Interface Linking Edges that you want to retrieve

Return type

Array of InterfaceLinkingEdge objects

Example

To make an array of InterfaceLinkingEdge objects for all of the Interface Linking Edges in model m flagged with f

```
var I_LE = InterfaceLinkingEdge.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the InterfaceLinkingEdge object for a Interface Linking Edge ID.

Arguments

Name	Type	Description
Model	Model	Model to find the Interface Linking Edge in
number	integer	number of the Interface Linking Edge you want the InterfaceLinkingEdge object for

Return type

InterfaceLinkingEdge object (or null if Interface Linking Edge does not exist).

Example

To get the InterfaceLinkingEdge object for Interface Linking Edge 100 in model m

```
var I_LE = InterfaceLinkingEdge.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a InterfaceLinkingEdge property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the InterfaceLinkingEdge.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	Interface Linking Edge property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if InterfaceLinkingEdge property I_LE.example is a parameter:

```
Options.property_parameter_names = true;
if (I_LE.GetParameter(I_LE.example) ) do_something...
Options.property_parameter_names = false;
```

To check if InterfaceLinkingEdge property I_LE.example is a parameter by using the GetParameter method:

```
if (I_LE.ViewParameters().GetParameter(I_LE.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this Interface Linking Edge (*INTERFACE_LINKING_EDGE). **Note that a carriage return is not added**. See also InterfaceLinkingEdge.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Interface Linking Edge m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Interface Linking Edge. **Note that a carriage return is not added**. See also InterfaceLinkingEdge.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Interface Linking Edge l:

```
var cards = l.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last Interface Linking Edge in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last Interface Linking Edge in

Return type

InterfaceLinkingEdge object (or null if there are no Interface Linking Edges in the model).

Example

To get the last Interface Linking Edge in model m:

```
var I_LE = InterfaceLinkingEdge.Last(m);
```

Next()

Description

Returns the next Interface Linking Edge in the model.

Arguments

No arguments

Return type

InterfaceLinkingEdge object (or null if there are no more Interface Linking Edges in the model).

Example

To get the Interface Linking Edge in model m after Interface Linking Edge I_LE:

```
var I_LE = I_LE.Next();
```

Previous()

Description

Returns the previous Interface Linking Edge in the model.

Arguments

No arguments

Return type

InterfaceLinkingEdge object (or null if there are no more Interface Linking Edges in the model).

Example

To get the Interface Linking Edge in model m before Interface Linking Edge I_LE:

```
var I_LE = I_LE.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select Interface Linking Edges using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting Interface Linking Edges
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only Interface Linking Edges from that model can be selected. If the argument is a Flag then only Interface Linking Edges that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any Interface Linking Edges can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of Interface Linking Edges selected or null if menu cancelled

Example

To select Interface Linking Edges from model m, flagging those selected with flag f, giving the prompt 'Select Interface Linking Edges':

```
InterfaceLinkingEdge.Select(f, 'Select Interface Linking Edges', m);
```

To select Interface Linking Edges, flagging those selected with flag f but limiting selection to Interface Linking Edges flagged with flag l, giving the prompt 'Select Interface Linking Edges':

```
InterfaceLinkingEdge.Select(f, 'Select Interface Linking Edges', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the Interface Linking Edge.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the Interface Linking Edge

Return type

No return value

Example

To set flag f for Interface Linking Edge I_LE:

```
I_LE.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of Interface Linking Edges in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing Interface Linking Edges should be counted. If false or omitted referenced but undefined Interface Linking Edges will also be included in the total.

Return type

number of Interface Linking Edges

Example

To get the total number of Interface Linking Edges in model m:

```
var total = InterfaceLinkingEdge.Total(m);
```

UnflagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Unsets a defined flag on all of the Interface Linking Edges in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all Interface Linking Edges will be unset in
flag	Flag	Flag to unset on the Interface Linking Edges

Return type

No return value

Example

To unset the flag f on all the Interface Linking Edges in model m:

```
InterfaceLinkingEdge.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

InterfaceLinkingEdge object.

Example

To check if InterfaceLinkingEdge property I_LE.example is a parameter by using the InterfaceLinkingEdge.GetParameter() method:

```
if (I_LE.ViewParameters().GetParameter(I_LE.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for Interface Linking Edge. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for Interface Linking Edge I_LE:

```
I_LE.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this Interface Linking Edge.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for Interface Linking Edge I_LE:

```
var xrefs = I_LE.Xrefs();
```

---

toString()

Description

Creates a string containing the Interface Linking Edge data in keyword format. Note that this contains the keyword header and the keyword cards. See also InterfaceLinkingEdge.Keyword() and InterfaceLinkingEdge.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Interface Linking Edge l in keyword format

```
var s = l.toString();
```

InterfaceSpringback class

The InterfaceSpringback class gives you access to interface springback cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetExcludeKeyword(idx*[integer]*)
- GetNodalPoint(npt*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveExcludeKeyword(idx*[integer]*)
- RemoveNodalPoint(npt*[integer]*)
- SetExcludeKeyword(keystr*[string]*, index (optional)*[integer]*)
- SetFlag(flag*[Flag]*)
- SetNodalPoint(npt*[integer]*, nid*[integer]*, tc*[real]*, rc*[real]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

InterfaceSpringback constants

Constants for Types of Keyword

Name	Description
InterfaceSpringback.EXCLUDE	INTERFACE is *INTERFACE_SPRINGBACK_EXCLUDE.
InterfaceSpringback.LSDYNA	INTERFACE is *INTERFACE_SPRINGBACK_LSDYNA.

InterfaceSpringback.NASTRAN	INTERFACE is *INTERFACE_SPRINGBACK_NASTRAN.
InterfaceSpringback.NIKE3D	INTERFACE is *INTERFACE_SPRINGBACK_NIKE3D.
InterfaceSpringback.SEAMLESS	INTERFACE is *INTERFACE_SPRINGBACK_SEAMLESS.

InterfaceSpringback properties

Name	Type	Description
cflag	integer	Output contact state.
exists	logical	true if interface springback exists, false if referred to but not defined. (read only)
fsplit	integer	Flag for splitting of the dynain file (0 - One file, 1 - Two files.). Used for OPTCARD field.
ftensr	integer	Flag for dumping tensor data from the element history variables into the dynain file (0/1).
ftype	integer	Filetype (0-3, 10-12).
include	integer	The Include file number that the interface springback is in.
intstrn	integer	Output of strains at all integration points of shell element is requested.
model	integer	The Model number that the interface springback is in.
ncyc	integer	Number of process cycles. Used for OPTCARD field.
ndflag	integer	Flag to dump nodes into dynain file.
nexclude	integer	gives the number of excluded keywords. Needed only for InterfaceSpringback.EXCLUDE.
nnodes	integer	gives the number of nodal points constrained for this keyword. (read_only)
nothickness	logical	true if _NOTHICKNESS (option2) is set. _NOTHICKNESS can be used only for InterfaceSpringback.LSDYNA or InterfaceSpringback.NASTRAN.
nshv	integer	Num additional Shell/Solid history variables number.
nthhsv	integer	Number of thermal history variables.
optcard	logical	Whether to have a OPTCARD. Can be true or false.
psid	integer	Part set ID for springback.
sldo	integer	Output of solid element data as 0 - *ELEMENT_SOLID, 1- *ELEM_SOLID_ORTHO. Used for OPTCARD field.
type	integer	gives the type of InterfaceSpringback object. (read only)

Detailed Description

The InterfaceSpringback class allows you to create, modify, edit and manipulate interface springback cards. See the documentation below for more details.

Constructor

new InterfaceSpringback(Model*[Model]*, Type*[constant]*, psid (optional) *[integer]*, nshv (optional)*[integer]*, ftype (optional)*[integer]*, ftensr (optional)*[integer]*, nthhsv (optional)*[integer]*, intstrn (optional)*[integer]*, optcard (optional)*[boolean]*, sldo (optional)*[integer]*, ncyc (optional)*[integer]*, fsplit (optional)*[integer]*, ndflag (optional)*[integer]*, cflag (optional)*[integer]*)

Description

Create a new InterfaceSpringback object.

Arguments

Name	Type	Description
Model	Model	Model that interface springback will be created in
Type	constant	Specify the type of InterfaceSpringback (Can be InterfaceSpringback.NIKE3D or InterfaceSpringback.LSDYNA or InterfaceSpringback.NASTRAN or InterfaceSpringback.SEAMLESS)
psid (optional)	integer	Part set ID for springback.
nshv (optional)	integer	Num additional Shell/Solid history variables number.
ftype (optional)	integer	Filetype (0-3, 10-12).
ftensr (optional)	integer	Flag for dumping tensor data from the element history variables into the dynain file (0/1).
nthhsv (optional)	integer	Number of thermal history variables.
intstrn (optional)	integer	Output of strains at all integration points of shell element is requested.
optcard (optional)	boolean	Whether to have an optional card. Can be true or false.
sldo (optional)	integer	Output of solid element data as 0 - *ELEMENT_SOLID, 1- *ELEM_SOLID_ORTHO. Used only for optional card.
ncyc (optional)	integer	Number of process cycles. Used only for optional card.
fsplit (optional)	integer	Flag for splitting of the dynain file (0 - One file, 1 - Two files.). Used only for optional card.
ndflag (optional)	integer	Flag to dump nodes into dynain file.
cflag (optional)	integer	Output contact state.

Return type

InterfaceSpringback object

Example

To create a new interface springback in model m, type LSDYNA, part set id 100:

```
var i_s = new InterfaceSpringback(m, InterfaceSringback.LSDYNA, 100);
```

new InterfaceSpringback(Model*[Model]*, Type*[constant]*, keylist (optional)*[Array of strings]*)

Description

Create a new InterfaceSpringback object with InterfaceSpringback.EXCLUDE option.

Arguments

Name	Type	Description
Model	Model	Model that interface springback will be created in
Type	constant	Specify the type of InterfaceSpringback (Should be InterfaceSpringback.EXCLUDE)

keylist (optional)	Array of strings	List of keywords to be excluded.

Return type

InterfaceSpringback object with InterfaceSpringback.EXCLUDE option.

Example

To create a new interface springback in model m, type EXCLUDE and keylist array keys:

```
var i_s = new InterfaceSpringback(m, InterfaceSringback.EXCLUDE, keys);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a interface springback.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the interface springback

Return type

No return value

Example

To associate comment c to the interface springback i_s:

```
i_s.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse interface springback i_s:

```
i_s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the interface springback.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the interface springback

Return type

No return value

Example

To clear flag f for interface springback i_s:

```
i_s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the interface springback.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

InterfaceSpringback object

Example

To copy interface springback i_s into interface springback z:

```
var z = i_s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an InterfaceSpringback definition.

Arguments

Name	Type	Description
Model	Model	Model that the InterfaceSpringback will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

InterfaceSpringback object (or null if not made)

Example

To start creating an ifce_sbak in model m:

```
var i_s = InterfaceSpringback.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a interface springback.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the interface springback

Return type

No return value

Example

To detach comment c from the interface springback i_s:

```
i_s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit interface springback i_s:

```
i_s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for interface springback. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give

details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for interface springback i_s:

```
i_s.Error("My custom error");
```

First(Model[*Model*]) [static]

Description

Returns the first interface springback in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first interface springback in

Return type

InterfaceSpringback object (or null if there are no interface springbacks in the model).

Example

To get the first interface springback in model m:

```
var i_s = InterfaceSpringback.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the interface springbacks in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all interface springbacks will be flagged in
flag	Flag	Flag to set on the interface springbacks

Return type

No return value

Example

To flag all of the interface springbacks with flag f in model m:

```
InterfaceSpringback.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the interface springback is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the interface springback

Return type

true if flagged, false if not.

Example

To check if interface springback i_s has flag f set on it:

```
if (i_s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each interface springback in the model.
Note that ForEach has been designed to make looping over interface springbacks as fast as possible and so has some limitations.
Firstly, a single temporary InterfaceSpringback object is created and on each function call it is updated with the current interface springback data. This means that you should not try to store the InterfaceSpringback object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new interface springbacks inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all interface springbacks are in
func	function	Function to call for each interface springback
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the interface springbacks in model m:

```
InterfaceSpringback.ForEach(m, test);
function test(i_s)
{
// i_s is InterfaceSpringback object
}
```

To call function test for all of the interface springbacks in model m with optional object:

```
var data = { x:0, y:0 };
InterfaceSpringback.ForEach(m, test, data);
function test(i_s, extra)
{
// i_s is InterfaceSpringback object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of InterfaceSpringback objects for all of the interface springbacks in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interface springbacks from

Return type

Array of InterfaceSpringback objects

Example

To make an array of InterfaceSpringback objects for all of the interface springbacks in model m

```
var i_s = InterfaceSpringback.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a interface springback.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the interface springback i_s:

```
var comm_array = i_s.GetComments();
```

GetExcludeKeyword(idx*[integer]*)

Description

Returns the keyword string excluded at given index in Keyword list. Needed only for InterfaceSpringback.EXCLUDE.

Arguments

Name	Type	Description
idx	integer	The index in Keyword list you want the Keyword string for. **Note that indices start at 0, not 1**.

Return type

A Keyword string at index "idx" from excluded keyword list .

Example

To get the 3rd Keyword string in Interface Springback i_s:

```
if (i_s.nexclude >= 3)
{
    var keyword = i_s.GetExcludeKeyword(2);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of InterfaceSpringback objects for all of the flagged interface springbacks in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get interface springbacks from
flag	Flag	Flag set on the interface springbacks that you want to retrieve

Return type

Array of InterfaceSpringback objects

Example

To make an array of InterfaceSpringback objects for all of the interface springbacks in model m flagged with f

```
var i_s = InterfaceSpringback.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the InterfaceSpringback object for a interface springback ID.

Arguments

Name	Type	Description
Model	Model	Model to find the interface springback in
number	integer	number of the interface springback you want the InterfaceSpringback object for

Return type

InterfaceSpringback object (or null if interface springback does not exist).

Example

To get the InterfaceSpringback object for interface springback 100 in model m

```
var i_s = InterfaceSpringback.GetFromID(m, 100);
```

GetNodalPoint(npt*[integer]*)

Description

Returns the data for nodal point constrained for *INTERFACE_SPRINGBACK.

Arguments

Name	Type	Description
npt	integer	The nodal point you want the data for. **Note that nodal points start at 0, not 1**.

Return type

An array containing the Node id, translational constraint (TC) and rotational constraint (RC) constants.

Example

To get the nodal point data for the 3rd nodal constraint for Interface Springback i_s:

```
if (i_s.nnodes >= 3)
{
    var npt_data = i_s.GetNodalPoint(2);
}
```

GetParameter(prop*[string]*)

Description

Checks if a InterfaceSpringback property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the InterfaceSpringback.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	interface springback property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if InterfaceSpringback property i_s.example is a parameter:

```
Options.property_parameter_names = true;
if (i_s.GetParameter(i_s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if InterfaceSpringback property i_s.example is a parameter by using the GetParameter method:

```
if (i_s.ViewParameters().GetParameter(i_s.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this Interface Springback (*INTERFACE_SPRINGBACK_xxxx_xxxx) **Note that a carriage return is not added**. See also InterfaceSpringback.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for InterfaceSpringback i_s:

```
var key = i_s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the InterfaceSpringback. **Note that a carriage return is not added**. See also InterfaceSpringback.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for InterfaceSpringback i_s:

```
var cards = i_s.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last interface springback in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last interface springback in

Return type

InterfaceSpringback object (or null if there are no interface springbacks in the model).

Example

To get the last interface springback in model m:

```
var i_s = InterfaceSpringback.Last(m);
```

Next()

Description

Returns the next interface springback in the model.

Arguments

No arguments

Return type

InterfaceSpringback object (or null if there are no more interface springbacks in the model).

Example

To get the interface springback in model m after interface springback i_s:

```
var i_s = i_s.Next();
```

Previous()

Description

Returns the previous interface springback in the model.

Arguments

No arguments

Return type

InterfaceSpringback object (or null if there are no more interface springbacks in the model).

Example

To get the interface springback in model m before interface springback i_s:

```
var i_s = i_s.Previous();
```

RemoveExcludeKeyword(idx*[integer]*)

Description

Removes the keyword string excluded at given index in Keyword list. Needed only for InterfaceSpringback.EXCLUDE

Arguments

Name	Type	Description
idx	integer	The index in Keyword list you removed. **Note that indices start at 0, not 1**.

Return type

No return value.

Example

To remove the 3rd Keyword string in Interface Springback i_s:

```
if (i_s.nexclude >= 3)
{
    var keyword = i_s.RemoveExcludeKeyword(2);
}
```

RemoveNodalPoint(npt*[integer]*)

Description

Removes the nodal point for constrained node for *INTERFACE_SPRINGBACK.

Arguments

Name	Type	Description
npt	integer	The nodal point you want to remove. **Note that nodal points start at 0, not 1**.

Return type

No return value.

Example

To remove the nodal point for the 3rd node for InterfaceSpringback i_s:

```
i_s.RemoveNodalPoint(2);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select interface springbacks using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting interface springbacks
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only interface springbacks from that model can be selected. If the argument is a Flag then only interface springbacks that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any interface springbacks can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of interface springbacks selected or null if menu cancelled

Example

To select interface springbacks from model m, flagging those selected with flag f, giving the prompt 'Select interface springbacks':

```
InterfaceSpringback.Select(f, 'Select interface springbacks', m);
```

To select interface springbacks, flagging those selected with flag f but limiting selection to interface springbacks flagged with flag l, giving the prompt 'Select interface springbacks':

```
InterfaceSpringback.Select(f, 'Select interface springbacks', l);
```

SetExcludeKeyword(keystr*[string]*, index (optional)*[integer]*)

Description

Sets a keyword string to be excluded. Adds a new keyword if index value is not given, else replaces the keyword string at given index. **Note that indices start at 0, not 1**. Needed only for InterfaceSpringback.EXCLUDE

Arguments

Name	Type	Description
keystr	string	The keyword string you want to be excluded.
index (optional)	integer	The index at which keyword string should be set.

Return type

No return value.

Example

To set a keyword string at index 3 to be excluded for InterfaceSpringback i_s:

```
if(i_s.nexclude >= 4)
    i_s.SetExcludeKeyword("ELEMENT_SHELL", 3);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the interface springback.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the interface springback

Return type

No return value

Example

To set flag f for interface springback i_s:

```
i_s.SetFlag(f);
```

SetNodalPoint(npt*[integer]*, nid*[integer]*, tc*[real]*, rc*[real]*)

Description

Sets the nodal point data for a node in *INTERFACE_SPRINGBACK.

Arguments

Name	Type	Description
npt	integer	The nodal point you want to set the data for. **Note that nodal points start at 0, not 1**.
nid	integer	Node ID for the nodal point.
tc	real	Translational constraint constant of the nodal point. (0-7)
rc	real	Rotational constraint constant of the nodal point. (0-7)

Return type

No return value.

Example

To set the nodal data for the 3rd nodal point to node 1, tc 2 and rc 4, for InterfaceSpringback i_s:

```
i_s.SetNodalPoint(2, 1, 2, 4);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of interface springbacks in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing interface springbacks should be counted. If false or omitted referenced but undefined interface springbacks will also be included in the total.

Return type

number of interface springbacks

Example

To get the total number of interface springbacks in model m:

```
var total = InterfaceSpringback.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the interface springbacks in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all interface springbacks will be unset in
flag	Flag	Flag to unset on the interface springbacks

Return type

No return value

Example

To unset the flag f on all the interface springbacks in model m:

```
InterfaceSpringback.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

InterfaceSpringback object.

Example

To check if InterfaceSpringback property i_s.example is a parameter by using the InterfaceSpringback.GetParameter() method:

```
if (i_s.ViewParameters().GetParameter(i_s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for interface springback. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for interface springback i_s:

```
i_s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this interface springback.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for interface springback i_s:

```
var xrefs = i_s.Xrefs();
```

---

toString()

Description

Creates a string containing the InterfaceSpringback data in keyword format. Note that this contains the keyword header and the keyword cards. See also InterfaceSpringback.Keyword() and InterfaceSpringback.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for InterfaceSpringback i_s in keyword format

```
var i_str = i_s.toString();
```

LoadBeam class

The LoadBeam class gives you access to define load beam cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadBeam constants

Name	Description
LoadBeam.ELEMENT	Load is *LOAD_BEAM_ELEMENT.
LoadBeam.SET	LOAD is *LOAD_BEAM_SET.

LoadBeam properties

Name	Type	Description
dal	integer	Direction of applied load. 1 for r-axis, 2 for s-axis or 3 for t-axis of beam.
eid	integer	NodeBeam ID or beam set ID. The esid property is an alternative name for this.
esid	integer	NodeBeam ID or beam set ID. The eid property is an alternative name for this.
exists	logical	true if load beam exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the load beam is in.
lcid	integer	Curve ID or function ID
model	integer	The Model number that the load beam is in.
sf	real	Load curve scale factor
type	constant	The Load Beam type. Can be LoadBeam.ELEMENT or LoadBeam.SET.

Detailed Description

The LoadBeam class allows you to create, modify, edit and manipulate load beam cards. See the documentation below for more details.

Constructor

new LoadBeam(Model*[Model]*, type*[constant]*, eid/esid*[integer]*, dal*[integer]*, lcid*[integer]*, sf (optional)*[real]*)

Description

Create a new LoadBeam object.

Arguments

Name	Type	Description
Model	Model	Model that load beam will be created in
type	constant	Specify the type of load beam (Can be LoadBeam.ELEMENT or LoadBeam.SET)
eid/esid	integer	Beam ID or beam set ID
dal	integer	Direction of applied load. 1 for r-axis, 2 for s-axis or 3 for t-axis of beam.
lcid	integer	Curve ID
sf (optional)	real	Load curve scale factor

Return type

LoadBeam object

Example

To create a new load beam in model m, of type SET, with beam set 100, load parallel to s-axis, loadcurve 9 and a scale factor of 0.5:

```
var lb = new LoadBeam(m, LoadBeam.SET, 100, 2, 9, 0.5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load beam

Return type

No return value

Example

To associate comment c to the load beam lb:

```
lb.AssociateComment(c);
```

Blank()

Description

Blanks the load beam

Arguments

No arguments

Return type

No return value

Example

To blank load beam lb:

```
lb.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load beams will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load beams in model m:

```
LoadBeam.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load beams will be blanked in
flag	Flag	Flag set on the load beams that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load beams in model m flagged with f:

```
LoadBeam.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load beam is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load beam lb is blanked:

```
if (lb.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse load beam lb:

```
lb.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load beam.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load beam

Return type

No return value

Example

To clear flag f for load beam lb:

```
lb.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load beam.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadBeam object

Example

To copy load beam lb into load beam z:

```
var z = lb.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a load beam definition.

Arguments

Name	Type	Description
Model	Model	Model that the load beam will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

LoadBeam object (or null if not made)

Example

To start creating a load beam definition in model m:

```
var lb = LoadBeam.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load beam.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load beam

Return type

No return value

Example

To detach comment c from the load beam lb:

```
lb.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit load beam lb:

```
lb.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load beam lb:

```
lb.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load beam in

Return type

LoadBeam object (or null if there are no load beams in the model).

Example

To get the first load beam in model m:

```
var lb = LoadBeam.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load beams in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load beams will be flagged in
flag	Flag	Flag to set on the load beams

Return type

No return value

Example

To flag all of the load beams with flag f in model m:

```
LoadBeam.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load beam is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load beam

Return type

true if flagged, false if not.

Example

To check if load beam lb has flag f set on it:

```
if (lb.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load beam in the model.
Note that ForEach has been designed to make looping over load beams as fast as possible and so has some limitations.
Firstly, a single temporary LoadBeam object is created and on each function call it is updated with the current load beam data. This means that you should not try to store the LoadBeam object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load beams inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load beams are in
func	function	Function to call for each load beam
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load beams in model m:

```
LoadBeam.ForEach(m, test);
function test(lb)
{
// lb is LoadBeam object
}
```

To call function test for all of the load beams in model m with optional object:

```
var data = { x:0, y:0 };
LoadBeam.ForEach(m, test, data);
function test(lb, extra)
{
// lb is LoadBeam object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of LoadBeam objects for all of the load beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load beams from

Return type

Array of LoadBeam objects

Example

To make an array of LoadBeam objects for all of the load beams in model m

```
var lb = LoadBeam.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load beam.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load beam lb:

```
var comm_array = lb.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadBeam objects for all of the flagged load beams in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load beams from
flag	Flag	Flag set on the load beams that you want to retrieve

Return type

Array of LoadBeam objects

Example

To make an array of LoadBeam objects for all of the load beams in model m flagged with f

```
var lb = LoadBeam.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadBeam object for a load beam ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load beam in
number	integer	number of the load beam you want the LoadBeam object for

Return type

LoadBeam object (or null if load beam does not exist).

Example

To get the LoadBeam object for load beam 100 in model m

```
var lb = LoadBeam.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadBeam property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadBeam.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load beam property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadBeam property lb.example is a parameter:

```
Options.property_parameter_names = true;
if (lb.GetParameter(lb.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadBeam property lb.example is a parameter by using the GetParameter method:

```
if (lb.ViewParameters().GetParameter(lb.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this load beam (*LOAD_BEAM_xxxx). **Note that a carriage return is not added**. See also LoadBeam.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for load beam lb:

```
var key = lb.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the load beam. **Note that a carriage return is not added**. See also LoadBeam.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for load beam lb:

```
var cards = lb.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last load beam in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load beam in

Return type

LoadBeam object (or null if there are no load beams in the model).

Example

To get the last load beam in model m:

```
var lb = LoadBeam.Last(m);
```

Next()

Description

Returns the next load beam in the model.

Arguments

No arguments

Return type

LoadBeam object (or null if there are no more load beams in the model).

Example

To get the load beam in model m after load beam lb:

```
var lb = lb.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load beam.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load beams from that model can be picked. If the argument is a Flag then only load beams that are flagged with *limit* can be selected. If omitted, or null, any load beams from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadBeam object (or null if not picked)

Example

To pick a load beam from model m giving the prompt 'Pick load beam from screen':

```
var lb = LoadBeam.Pick('Pick load beam from screen', m);
```

Previous()

Description

Returns the previous load beam in the model.

Arguments

No arguments

Return type

LoadBeam object (or null if there are no more load beams in the model).

Example

To get the load beam in model m before load beam lb:

```
var lb = lb.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load beams using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load beams
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load beams from that model can be selected. If the argument is a Flag then only load beams that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load beams can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load beams selected or null if menu cancelled

Example

To select load beams from model m, flagging those selected with flag f, giving the prompt 'Select load beams':

```
LoadBeam.Select(f, 'Select load beams', m);
```

To select load beams, flagging those selected with flag f but limiting selection to load beams flagged with flag l, giving the prompt 'Select load beams':

```
LoadBeam.Select(f, 'Select load beams', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load beam.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load beam

Return type

No return value

Example

To set flag f for load beam lb:

```
lb.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load beam. The load beam will be sketched until you either call LoadBeam.Unsketch(), LoadBeam.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load beam is sketched. If omitted redraw is true. If you want to sketch several load beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load beam lb:

```
lb.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load beams in the model. The load beams will be sketched until you either call LoadBeam.Unsketch(), LoadBeam.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load beams will be sketched in
flag	Flag	Flag set on the load beams that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load beams are sketched. If omitted redraw is true. If you want to sketch flagged load beams several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load beams flagged with flag in model m:

```
LoadBeam.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load beams should be counted. If false or omitted referenced but undefined load beams will also be included in the total.

Return type

number of load beams

Example

To get the total number of load beams in model m:

```
var total = LoadBeam.Total(m);
```

Unblank()

Description

Unblanks the load beam

Arguments

No arguments

Return type

No return value

Example

To unblank load beam lb:

```
lb.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load beams in the model.

Arguments

Name	Type	Description

Model	Model	Model that all load beams will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load beams in model m:

```
LoadBeam.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load beams will be unblanked in
flag	Flag	Flag set on the load beams that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load beams in model m flagged with f:

```
LoadBeam.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load beams will be unset in
flag	Flag	Flag to unset on the load beams

Return type

No return value

Example

To unset the flag f on all the load beams in model m:

```
LoadBeam.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load beam.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load beam is unsketched. If omitted redraw is true. If you want to unsketch several load beams and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load beam lb:

```
lb.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load beams.

Arguments

Name	Type	Description
Model	Model	Model that all load beams will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load beams in model m:

```
LoadBeam.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load beams in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load beams will be unsketched in
flag	Flag	Flag set on the load beams that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the load beams are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load beams flagged with flag in model m:

```
LoadBeam.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadBeam object.

Example

To check if LoadBeam property lb.example is a parameter by using the LoadBeam.GetParameter() method:

```
if (lb.ViewParameters().GetParameter(lb.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load beam. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load beam lb:

```
lb.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load beam.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load beam lb:

```
var xrefs = lb.Xrefs();
```

---

toString()

Description

Creates a string containing the load beam data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadBeam.Keyword() and LoadBeam.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for load beam lb in keyword format

```
var s = lb.toString();
```

LoadBody class

The LoadBody class gives you access to *LOAD_BODY cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

LoadBody properties

Name	Type	Description
parts	Object	*LOAD_BODY_PARTS card
rx	Object	*LOAD_BODY_RX card
ry	Object	*LOAD_BODY_RY card
rz	Object	*LOAD_BODY_RZ card
vector	Object	*LOAD_BODY_VECTOR card
x	Object	*LOAD_BODY_X card
y	Object	*LOAD_BODY_Y card
z	Object	*LOAD_BODY_Z card

Properties for *LOAD_BODY

Name	Type	Description
cid	integer	Coordinate system ID (not _PARTS)
exists	logical	true if LoadBody card exists
include	integer	The Include file number that the LoadBody card is in.
lcid	integer	Load curve ID (not _PARTS)
lciddr	integer	Load curve ID for dynamic relaxation (not _PARTS)
psid	integer	Part set id (_PARTS only)
sf	real	Load curve scale factor (not _PARTS)
v1	real	X-component of Vector
v2	real	Y-component of Vector
v3	real	Z-component of Vector
xc	real	X centre of rotation (_RX, _RY and _RZ)
yc	real	Y centre of rotation (_RX, _RY and _RZ)
zc	real	Z centre of rotation (_RX, _RY and _RZ)

Detailed Description

The LoadBody class allows you to create, modify, edit and manipulate *LOAD_BODY cards. Unlike other classes there is no constructor and there are no functions. Instead a LoadBody object is available as the loadBody property of a Model object. This object allows you to access all of the *LOAD_BODY cards.
For example, to activate *LOAD_BODY_X in model m and set lcid to 1.

```
m.loadBody.x.exists = true;
m.loadBody.x.lcid = 1;
```

See the properties for more details.

LoadBodyGeneralized class

The LoadBodyGeneralized class gives you access to define load body generalized cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- BlankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- First(Model *[Model]*)
- FlagAll(Model *[Model]*, flag *[Flag]*)
- ForEach(Model *[Model]*, func *[function]*, extra (optional) *[any]*)
- GetAll(Model *[Model]*)
- GetFlagged(Model *[Model]*, flag *[Flag]*)
- GetFromID(Model *[Model]*, number *[integer]*)
- Last(Model *[Model]*)
- Pick(prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*, button text (optional) *[string]*)
- Select(flag *[Flag]*, prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*)
- SketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Total(Model *[Model]*, exists (optional) *[boolean]*)
- UnblankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnblankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- UnflagAll(Model *[Model]*, flag *[Flag]*)
- UnsketchAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnsketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)

Member functions

- AssociateComment(Comment *[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag *[Flag]*)
- Copy(range (optional) *[boolean]*)
- DetachComment(Comment *[Comment]*)
- Error(message *[string]*, details (optional) *[string]*)
- Flagged(flag *[Flag]*)
- GetComments()
- GetParameter(prop *[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag *[Flag]*)
- Sketch(redraw (optional) *[boolean]*)
- Unblank()
- Unsketch(redraw (optional) *[boolean]*)
- ViewParameters()
- Warning(message *[string]*, details (optional) *[string]*)
- Xrefs()
- toString()

LoadBodyGeneralized constants

Name	Description

LoadBodyGeneralized.NODE	Load is *LOAD_BODY_GENERALIZED.
LoadBodyGeneralized.SET_NODE	Load is *LOAD_BODY_GENERALIZED_SET_NODE.
LoadBodyGeneralized.SET_PART	LOAD is *LOAD_BODY_GENERALIZED_SET_PART.

LoadBodyGeneralized properties

Name	Type	Description
angtyp	string	Type of body loads
ax	real	Scale factor for acceleration in x-direction
ay	real	Scale factor for acceleration in y-direction
az	real	Scale factor for acceleration in z-direction
cid	integer	Coordinate system ID to define acceleration
drlcid	real	Curve ID for dynamic relaxation phase
exists	logical	true if load body generalized exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the load body generalized is in.
lcid	integer	Curve ID
model	integer	The Model number that the load body generalized is in.
n1	integer	Beginning Node ID for body force load or the node or Part set ID
n2	integer	Ending Node ID for body force load. Set to zero if a set ID is defined
omx	real	Scale factor for x-angular velocity or acceleration
omy	real	Scale factor for y-angular velocity or acceleration
omz	real	Scale factor for z-angular velocity or acceleration
type	constant	The Load Node type, can be LoadBodyGeneralized.NODE or LoadBodyGeneralized.SET_NODE or LoadBodyGeneralized.SET_PART.
xc	real	X-center of rotation
yc	real	Y-center of rotation
zc	real	Z-center of rotation

Detailed Description

The LoadBodyGeneralized class allows you to create, modify, edit and manipulate load body generalized cards. See the documentation below for more details.

Constructor

new LoadBodyGeneralized(Model*[Model]*, type*[constant]*, n1*[integer]*, n2*[integer]*, lcid*[integer]*, drlcid (optional)*[integer]*, xc (optional)*[real]*, yc (optional)*[real]*, zc (optional)*[real]*, ax (optional)*[real]*, ay (optional)*[real]*, az (optional)*[real]*, omx (optional)*[real]*, omy (optional)*[real]*, omz (optional)*[real]*, cid (optional)*[integer]*, angtyp (optional)*[string]*)

Description

Create a new LoadBodyGeneralized object.

Arguments

Name	Type	Description
Model	Model	Model that load body generalized will be created in
type	constant	Specify the type of load body generalized (Can be LoadBodyGeneralized.NODE or LoadBodyGeneralized.SET_NODE or LoadBodyGeneralized.SET_PART)
n1	integer	Beginning Node ID for body force load or the node or Part set ID
n2	integer	Ending Node ID for body force load. Set to zero if a set ID is defined
lcid	integer	Curve ID
drlcid (optional)	integer	Curve ID for dynamic relaxation phase
xc (optional)	real	X-center of rotation
yc (optional)	real	Y-center of rotation
zc (optional)	real	Z-center of rotation
ax (optional)	real	Scale factor for acceleration in x-direction
ay (optional)	real	Scale factor for acceleration in y-direction
az (optional)	real	Scale factor for acceleration in z-direction
omx (optional)	real	Scale factor for x-angular velocity or acceleration
omy (optional)	real	Scale factor for y-angular velocity or acceleration
omz (optional)	real	Scale factor for z-angular velocity or acceleration
cid (optional)	integer	Coordinate system ID to define acceleration
angtyp (optional)	string	Type of body loads

Return type

LoadBodyGeneralized object

Example

To create a new load body generalized in model m, of type SET_NODE, with LCID 9 and N2 is 2

```
var b = new LoadBodyGeneralized(m, LoadBodyGeneralized.SET_NODE, 100, 2, 9);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a load body generalized.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load body generalized

Return type

No return value

Example

To associate comment c to the load body generalized lbg:

```
lbg.AssociateComment(c);
```

Blank()

Description

Blanks the load body generalized

Arguments

No arguments

Return type

No return value

Example

To blank load body generalized lbg:

```
lbg.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load body generalizeds in model m:

```
LoadBodyGeneralized.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load body generalizeds will be blanked in

flag	Flag	Flag set on the load body generalizeds that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load body generalizeds in model m flagged with f:

```
LoadBodyGeneralized.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load body generalized is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load body generalized lbg is blanked:

```
if (lbg.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load body generalized.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load body generalized

Return type

No return value

Example

To clear flag f for load body generalized lbg:

```
lbg.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load body generalized.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadBodyGeneralized object

Example

To copy load body generalized lbg into load body generalized z:

```
var z = lbg.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load body generalized.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load body generalized

Return type

No return value

Example

To detach comment c from the load body generalized lbg:

```
lbg.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load body generalized. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load body generalized lbg:

```
lbg.Error("My custom error");
```

First(Model[*Model*]) [static]

Description

Returns the first load body generalized in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load body generalized in

Return type

LoadBodyGeneralized object (or null if there are no load body generalizeds in the model).

Example

To get the first load body generalized in model m:

```
var lbg = LoadBodyGeneralized.First(m);
```

FlagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Flags all of the load body generalizeds in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds will be flagged in
flag	Flag	Flag to set on the load body generalizeds

Return type

No return value

Example

To flag all of the load body generalizeds with flag f in model m:

```
LoadBodyGeneralized.FlagAll(m, f);
```

Flagged(flag[*Flag*])

Description

Checks if the load body generalized is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load body generalized

Return type

true if flagged, false if not.

Example

To check if load body generalized lbg has flag f set on it:

```
if (lbg.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load body generalized in the model.
Note that ForEach has been designed to make looping over load body generalizeds as fast as possible and so has some limitations.
Firstly, a single temporary LoadBodyGeneralized object is created and on each function call it is updated with the current load body generalized data. This means that you should not try to store the LoadBodyGeneralized object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load body generalizeds inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds are in
func	function	Function to call for each load body generalized
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load body generalizeds in model m:

```
LoadBodyGeneralized.ForEach(m, test);
function test(lbg)
{
// lbg is LoadBodyGeneralized object
}
```

To call function test for all of the load body generalizeds in model m with optional object:

```
var data = { x:0, y:0 };
LoadBodyGeneralized.ForEach(m, test, data);
function test(lbg, extra)
{
// lbg is LoadBodyGeneralized object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of LoadBodyGeneralized objects for all of the load body generalizeds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load body generalizeds from

Return type

Array of LoadBodyGeneralized objects

Example

To make an array of LoadBodyGeneralized objects for all of the load body generalizeds in model m

```
var lbg = LoadBodyGeneralized.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load body generalized.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load body generalized lbg:

```
var comm_array = lbg.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadBodyGeneralized objects for all of the flagged load body generalizeds in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load body generalizeds from
flag	Flag	Flag set on the load body generalizeds that you want to retrieve

Return type

Array of LoadBodyGeneralized objects

Example

To make an array of LoadBodyGeneralized objects for all of the load body generalizeds in model m flagged with f

```
var lbg = LoadBodyGeneralized.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadBodyGeneralized object for a load body generalized ID.

Arguments

Name	Type	Description

Model	Model	Model to find the load body generalized in
number	integer	number of the load body generalized you want the LoadBodyGeneralized object for

Return type

LoadBodyGeneralized object (or null if load body generalized does not exist).

Example

To get the LoadBodyGeneralized object for load body generalized 100 in model m

```
var lbg = LoadBodyGeneralized.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadBodyGeneralized property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadBodyGeneralized.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load body generalized property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadBodyGeneralized property lbg.example is a parameter:

```
Options.property_parameter_names = true;
if (lbg.GetParameter(lbg.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadBodyGeneralized property lbg.example is a parameter by using the GetParameter method:

```
if (lbg.ViewParameters().GetParameter(lbg.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this load body generalized (*LOAD_NODE_xxxx). **Note that a carriage return is not added**. See also LoadBodyGeneralized.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for load body generalized m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the load body generalized. **Note that a carriage return is not added**. See also LoadBodyGeneralized.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for load body generalized l:

```
var cards = l.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last load body generalized in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load body generalized in

Return type

LoadBodyGeneralized object (or null if there are no load body generalizeds in the model).

Example

To get the last load body generalized in model m:

```
var lbg = LoadBodyGeneralized.Last(m);
```

Next()

Description

Returns the next load body generalized in the model.

Arguments

No arguments

Return type

LoadBodyGeneralized object (or null if there are no more load body generalizeds in the model).

Example

To get the load body generalized in model m after load body generalized lbg:

```
var lbg = lbg.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load body generalized.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load body generalizeds from that model can be picked. If the argument is a Flag then only load body generalizeds that are flagged with *limit* can be selected. If omitted, or null, any load body generalizeds from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadBodyGeneralized object (or null if not picked)

Example

To pick a load body generalized from model m giving the prompt 'Pick load body generalized from screen':

```
var lbg = LoadBodyGeneralized.Pick('Pick load body generalized from screen', m);
```

Previous()

Description

Returns the previous load body generalized in the model.

Arguments

No arguments

Return type

LoadBodyGeneralized object (or null if there are no more load body generalizeds in the model).

Example

To get the load body generalized in model m before load body generalized lbg:

```
var lbg = lbg.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load body generalizeds using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load body generalizeds
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load body generalizeds from that model can be selected. If the argument is a Flag then only load body generalizeds that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load body generalizeds can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load body generalizeds selected or null if menu cancelled

Example

To select load body generalizeds from model m, flagging those selected with flag f, giving the prompt 'Select load body generalizeds':

```
LoadBodyGeneralized.Select(f, 'Select load body generalizeds', m);
```

To select load body generalizeds, flagging those selected with flag f but limiting selection to load body generalizeds flagged with flag l, giving the prompt 'Select load body generalizeds':

```
LoadBodyGeneralized.Select(f, 'Select load body generalizeds', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load body generalized.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load body generalized

Return type

No return value

Example

To set flag f for load body generalized lbg:

```
lbg.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load body generalized. The load body generalized will be sketched until you either call LoadBodyGeneralized.Unsketch(), LoadBodyGeneralized.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load body generalized is sketched. If omitted redraw is true. If you want to sketch several load body generalizeds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load body generalized lbg:

```
lbg.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load body generalizeds in the model. The load body generalizeds will be sketched until you either call LoadBodyGeneralized.Unsketch(), LoadBodyGeneralized.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load body generalizeds will be sketched in
flag	Flag	Flag set on the load body generalizeds that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load body generalizeds are sketched. If omitted redraw is true. If you want to sketch flagged load body generalizeds several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load body generalizeds flagged with flag in model m:

```
LoadBodyGeneralized.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing load body generalizeds should be counted. If false or omitted referenced but undefined load body generalizeds will also be included in the total.

Return type

number of load body generalizeds

Example

To get the total number of load body generalizeds in model m:

```
var total = LoadBodyGeneralized.Total(m);
```

Unblank()

Description

Unblanks the load body generalized

Arguments

No arguments

Return type

No return value

Example

To unblank load body generalized lbg:

```
lbg.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load body generalizeds in model m:

```
LoadBodyGeneralized.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load body generalizeds will be unblanked in
flag	Flag	Flag set on the load body generalizeds that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load body generalizeds in model m flagged with f:

```
LoadBodyGeneralized.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load body generalizeds will be unset in
flag	Flag	Flag to unset on the load body generalizeds

Return type

No return value

Example

To unset the flag f on all the load body generalizeds in model m:

```
LoadBodyGeneralized.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load body generalized.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load body generalized is unsketched. If omitted redraw is true. If you want to unsketch several load body generalizeds and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load body generalized lbg:

```
lbg.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load body generalizeds.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load body generalizeds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load body generalizeds in model m:

```
LoadBodyGeneralized.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load body generalizeds in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load body generalizeds will be unsketched in
flag	Flag	Flag set on the load body generalizeds that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the load body generalizeds are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load body generalizeds flagged with flag in model m:

```
LoadBodyGeneralized.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadBodyGeneralized object.

Example

To check if LoadBodyGeneralized property lbg.example is a parameter by using the LoadBodyGeneralized.GetParameter() method:

```
if (lbg.ViewParameters().GetParameter(lbg.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load body generalized. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load body generalized lbg:

```
lbg.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load body generalized.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load body generalized lbg:

```
var xrefs = lbg.Xrefs();
```

---

toString()

Description

Creates a string containing the load body generalized data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadBodyGeneralized.Keyword() and LoadBodyGeneralized.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for load body generalized l in keyword format

```
var s = l.toString();
```

LoadGravity class

The LoadGravity class gives you access to define *LOAD_GRAVITY_PART cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadGravity constants

Name	Description
LoadGravity.PART	LOAD is *LOAD_GRAVITY_PART.

LoadGravity.SET_PART	LOAD is *LOAD_GRAVITY_PART_SET.

LoadGravity properties

Name	Type	Description
accel	real	Acceleration (will be multiplied by factor from curve)
dof	integer	Direction: enter 1, 2 or 3 for x, y, or z
exists	logical	true if LoadGravity exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the LoadGravity is in.
lc	integer	Curve ID. Load curve defining factor vs. time (or zero if STGA, STGR are defined)
lcdr	integer	Curve ID. Load curve defining factor vs. time during dynamic relaxation
model	integer	The Model number that the load gravity is in.
pid	integer	Part ID or Part set ID
stga	integer	Construction Stages ID at which part is added (optional)
stgr	integer	Construction Stages ID at which part is removed (optional)
type	constant	The Load Gravity type. Can be LoadGravity.PART or LoadGravity.SET_PART.

Detailed Description

The LoadGravity class allows you to create, modify, edit and manipulate *LOAD_GRAVITY_PART cards. See the documentation below for more details.

Constructor

new LoadGravity(Model*[Model]*, type*[constant]*, pid*[integer]*, dof*[integer]*, lc*[integer]*, accel*[real]*, lcdr*[integer]*, stga (optional)*[integer]*, stgr (optional)*[integer]*)

Description

Create a new LoadGravity object.

Arguments

Name	Type	Description
Model	Model	Model that LoadGravity will be created in
type	constant	Specify the type of LoadGravity (Can be LoadGravity.PART or LoadGravity.SET_PART
pid	integer	Part ID or Part set ID
dof	integer	Direction: enter 1, 2 or 3 for x, y or z
lc	integer	Curve ID. Load curve defining factor vs. time (or zero if STGA, STGR are defined)
accel	real	Acceleration (will be multiplied by factor from curve)
lcdr	integer	Curve ID. Load curve defining factor vs. time during dynamic relaxation
stga (optional)	integer	Construction Stage ID at which part is added (optional)
stgr (optional)	integer	Construction Stage ID at which part is removed (optional)

Return type

LoadGravity object

Example

To create a new load gravity in model m, of type SET, with dof 2, loadcurve 9, acceleration of 0.5, and lcdr 10

```
var lg = new LoadGravity(m, LoadGravity.PART, 100, 2, 9, 0.5, 10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load gravity.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load gravity

Return type

No return value

Example

To associate comment c to the load gravity lg:

```
lg.AssociateComment(c);
```

Blank()

Description

Blanks the load gravity

Arguments

No arguments

Return type

No return value

Example

To blank load gravity lg:

```
lg.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load gravitys in the model.

Arguments

Name	Type	Description

Model	Model	Model that all load gravitys will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load gravitys in model m:

```
LoadGravity.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load gravitys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load gravitys will be blanked in
flag	Flag	Flag set on the load gravitys that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load gravitys in model m flagged with f:

```
LoadGravity.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load gravity is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load gravity lg is blanked:

```
if (lg.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load gravity.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load gravity

Return type

No return value

Example

To clear flag f for load gravity lg:

```
lg.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load gravity.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadGravity object

Example

To copy load gravity lg into load gravity z:

```
var z = lg.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load gravity.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load gravity

Return type

No return value

Example

To detach comment c from the load gravity lg:

```
lg.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load gravity. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load gravity lg:

```
lg.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load gravity in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load gravity in

Return type

LoadGravity object (or null if there are no load gravitys in the model).

Example

To get the first load gravity in model m:

```
var lg = LoadGravity.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load gravitys in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load gravitys will be flagged in

flag	Flag	Flag to set on the load gravitys

Return type

No return value

Example

To flag all of the load gravitys with flag f in model m:

```
LoadGravity.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load gravity is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load gravity

Return type

true if flagged, false if not.

Example

To check if load gravity lg has flag f set on it:

```
if (lg.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load gravity in the model.
Note that ForEach has been designed to make looping over load gravitys as fast as possible and so has some limitations.
Firstly, a single temporary LoadGravity object is created and on each function call it is updated with the current load gravity data. This means that you should not try to store the LoadGravity object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load gravitys inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load gravitys are in
func	function	Function to call for each load gravity
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load gravitys in model m:

```
LoadGravity.ForEach(m, test);
function test(lg)
{
// lg is LoadGravity object
}
```

To call function test for all of the load gravitys in model m with optional object:

```
var data = { x:0, y:0 };
LoadGravity.ForEach(m, test, data);
function test(lg, extra)
{
// lg is LoadGravity object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of LoadGravity objects for all of the load gravitys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load gravitys from

Return type

Array of LoadGravity objects

Example

To make an array of LoadGravity objects for all of the load gravitys in model m

```
var lg = LoadGravity.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load gravity.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load gravity lg:

```
var comm_array = lg.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadGravity objects for all of the flagged load gravitys in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load gravitys from
flag	Flag	Flag set on the load gravitys that you want to retrieve

Return type

Array of LoadGravity objects

Example

To make an array of LoadGravity objects for all of the load gravitys in model m flagged with f

```
var lg = LoadGravity.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadGravity object for a load gravity ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load gravity in
number	integer	number of the load gravity you want the LoadGravity object for

Return type

LoadGravity object (or null if load gravity does not exist).

Example

To get the LoadGravity object for load gravity 100 in model m

```
var lg = LoadGravity.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadGravity property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadGravity.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load gravity property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadGravity property lg.example is a parameter:

```
Options.property_parameter_names = true;
if (lg.GetParameter(lg.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadGravity property lg.example is a parameter by using the GetParameter method:

```
if (lg.ViewParameters().GetParameter(lg.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this LoadGravity (*LOAD_GRAVITY_PART). **Note that a carriage return is not added**. See also LoadGravity.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for LoadGravity lg:

```
var key = lg.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the LoadGravity. **Note that a carriage return is not added**. See also LoadGravity.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for LoadGravity lg:

```
var cards = lg.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last load gravity in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load gravity in

Return type

LoadGravity object (or null if there are no load gravitys in the model).

Example

To get the last load gravity in model m:

```
var lg = LoadGravity.Last(m);
```

Next()

Description

Returns the next load gravity in the model.

Arguments

No arguments

Return type

LoadGravity object (or null if there are no more load gravitys in the model).

Example

To get the load gravity in model m after load gravity lg:

```
var lg = lg.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load gravity.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load gravitys from that model can be picked. If the argument is a Flag then only load gravitys that are flagged with *limit* can be selected. If omitted, or null, any load gravitys from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadGravity object (or null if not picked)

Example

To pick a load gravity from model m giving the prompt 'Pick load gravity from screen':

```
var lg = LoadGravity.Pick('Pick load gravity from screen', m);
```

Previous()

Description

Returns the previous load gravity in the model.

Arguments

No arguments

Return type

LoadGravity object (or null if there are no more load gravitys in the model).

Example

To get the load gravity in model m before load gravity lg:

```
var lg = lg.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load gravitys using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load gravitys
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load gravitys from that model can be selected. If the argument is a Flag then only load gravitys that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load gravitys can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load gravitys selected or null if menu cancelled

Example

To select load gravitys from model m, flagging those selected with flag f, giving the prompt 'Select load gravitys':

```
LoadGravity.Select(f, 'Select load gravitys', m);
```

To select load gravitys, flagging those selected with flag f but limiting selection to load gravitys flagged with flag l, giving the prompt 'Select load gravitys':

```
LoadGravity.Select(f, 'Select load gravitys', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load gravity.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load gravity

Return type

No return value

Example

To set flag f for load gravity lg:

```
lg.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load gravity. The load gravity will be sketched until you either call LoadGravity.Unsketch(), LoadGravity.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load gravity is sketched. If omitted redraw is true. If you want to sketch several load gravitys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load gravity lg:

```
lg.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load gravitys in the model. The load gravitys will be sketched until you either call LoadGravity.Unsketch(), LoadGravity.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load gravitys will be sketched in
flag	Flag	Flag set on the load gravitys that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load gravitys are sketched. If omitted redraw is true. If you want to sketch flagged load gravitys several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load gravitys flagged with flag in model m:

```
LoadGravity.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load gravitys in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load gravitys should be counted. If false or omitted referenced but undefined load gravitys will also be included in the total.

Return type

number of load gravitys

Example

To get the total number of load gravitys in model m:

```
var total = LoadGravity.Total(m);
```

Unblank()

Description

Unblanks the load gravity

Arguments

No arguments

Return type

No return value

Example

To unblank load gravity lg:

```
lg.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load gravitys in the model.

Arguments

Name	Type	Description

Model	Model	Model that all load gravitys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load gravitys in model m:

```
LoadGravity.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load gravitys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load gravitys will be unblanked in
flag	Flag	Flag set on the load gravitys that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load gravitys in model m flagged with f:

```
LoadGravity.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load gravitys in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load gravitys will be unset in
flag	Flag	Flag to unset on the load gravitys

Return type

No return value

Example

To unset the flag f on all the load gravitys in model m:

```
LoadGravity.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load gravity.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load gravity is unsketched. If omitted redraw is true. If you want to unsketch several load gravitys and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load gravity lg:

```
lg.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load gravitys.

Arguments

Name	Type	Description
Model	Model	Model that all load gravitys will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load gravitys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load gravitys in model m:

```
LoadGravity.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load gravitys in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load gravitys will be unsketched in
flag	Flag	Flag set on the load gravitys that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the load gravitys are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load gravitys flagged with flag in model m:

```
LoadGravity.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadGravity object.

Example

To check if LoadGravity property lg.example is a parameter by using the LoadGravity.GetParameter() method:

```
if (lg.ViewParameters().GetParameter(lg.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load gravity. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load gravity lg:

```
lg.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load gravity.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load gravity lg:

```
var xrefs = lg.Xrefs();
```

---

toString()

Description

Creates a string containing the LoadGravity data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadGravity.Keyword() and LoadGravity.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for LoadGravity lg in keyword format

```
var s = lg.toString();
```

LoadNode class

The LoadNode class gives you access to define load node cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadNode constants

Name	Description
LoadNode.POINT	Load is *LOAD_NODE_POINT.

LoadNode.SET	LOAD is *LOAD_NODE_SET.

LoadNode properties

Name	Type	Description
cid	integer	Coordinate system ID
dof	integer	Applicable degrees-of-freedom
exists	logical	true if load node exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the load node is in.
lcid	integer	Curve ID
m1	integer	Node 1 ID
m2	integer	Node 2 ID
m3	integer	Node 3 ID
model	integer	The Model number that the load node is in.
nid	integer	Node ID or node set ID
sf	real	Curve scale factor
type	constant	The Load Node type. Can be LoadNode.POINT or LoadNode.SET.

Detailed Description

The LoadNode class allows you to create, modify, edit and manipulate load node cards. See the documentation below for more details.

Constructor

new LoadNode(Model*[Model]*, type*[constant]*, nid*[integer]*, dof*[integer]*, lcid*[integer]*, sf (optional)*[real]*, cid (optional)*[integer]*, m1 (optional)*[integer]*, m2 (optional)*[integer]*, m3 (optional)*[integer]*)

Description

Create a new LoadNode object.

Arguments

Name	Type	Description
Model	Model	Model that load node will be created in
type	constant	Specify the type of load node (Can be LoadNode.POINT or LoadNode.SET)
nid	integer	Node ID or node set ID
dof	integer	Applicable degrees-of-freedom
lcid	integer	Curve ID
sf (optional)	real	Curve scale factor
cid (optional)	integer	Coordinate system ID
m1 (optional)	integer	Node 1 ID
m2 (optional)	integer	Node 2 ID
m3 (optional)	integer	Node 3 ID

Return type

LoadNode object

Example

To create a new load node in model m, of type SET, with loadcurve 9 and a scale factor of 0.5

```
var b = new LoadNode(m, LoadNode.SET, 100, 2, 9, 0.5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load node

Return type

No return value

Example

To associate comment c to the load node ln:

```
ln.AssociateComment(c);
```

Blank()

Description

Blanks the load node

Arguments

No arguments

Return type

No return value

Example

To blank load node ln:

```
ln.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load nodes in model m:

```
LoadNode.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load nodes will be blanked in
flag	Flag	Flag set on the load nodes that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load nodes in model m flagged with f:

```
LoadNode.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load node is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load node ln is blanked:

```
if (ln.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load node.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load node

Return type

No return value

Example

To clear flag f for load node ln:

```
ln.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load node.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadNode object

Example

To copy load node ln into load node z:

```
var z = ln.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load node

Return type

No return value

Example

To detach comment c from the load node ln:

```
ln.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load node ln:

```
ln.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load node in

Return type

LoadNode object (or null if there are no load nodes in the model).

Example

To get the first load node in model m:

```
var ln = LoadNode.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load nodes in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes will be flagged in

flag	Flag	Flag to set on the load nodes

Return type

No return value

Example

To flag all of the load nodes with flag f in model m:

```
LoadNode.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load node is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load node

Return type

true if flagged, false if not.

Example

To check if load node ln has flag f set on it:

```
if (ln.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load node in the model.
Note that ForEach has been designed to make looping over load nodes as fast as possible and so has some limitations.
Firstly, a single temporary LoadNode object is created and on each function call it is updated with the current load node data. This means that you should not try to store the LoadNode object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load nodes inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes are in
func	function	Function to call for each load node
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load nodes in model m:

```
LoadNode.ForEach(m, test);
function test(ln)
{
// ln is LoadNode object
}
```

To call function test for all of the load nodes in model m with optional object:

```
var data = { x:0, y:0 };
LoadNode.ForEach(m, test, data);
function test(ln, extra)
{
// ln is LoadNode object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of LoadNode objects for all of the load nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load nodes from

Return type

Array of LoadNode objects

Example

To make an array of LoadNode objects for all of the load nodes in model m

```
var ln = LoadNode.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load node.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load node ln:

```
var comm_array = ln.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadNode objects for all of the flagged load nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load nodes from
flag	Flag	Flag set on the load nodes that you want to retrieve

Return type

Array of LoadNode objects

Example

To make an array of LoadNode objects for all of the load nodes in model m flagged with f

```
var ln = LoadNode.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadNode object for a load node ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load node in
number	integer	number of the load node you want the LoadNode object for

Return type

LoadNode object (or null if load node does not exist).

Example

To get the LoadNode object for load node 100 in model m

```
var ln = LoadNode.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadNode property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadNode.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load node property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadNode property ln.example is a parameter:

```
Options.property_parameter_names = true;
if (ln.GetParameter(ln.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadNode property ln.example is a parameter by using the GetParameter method:

```
if (ln.ViewParameters().GetParameter(ln.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this load node (*LOAD_NODE_xxxx). **Note that a carriage return is not added**. See also LoadNode.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for load node m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the load node. **Note that a carriage return is not added**. See also LoadNode.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for load node l:

```
var cards = l.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last load node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load node in

Return type

LoadNode object (or null if there are no load nodes in the model).

Example

To get the last load node in model m:

```
var ln = LoadNode.Last(m);
```

Next()

Description

Returns the next load node in the model.

Arguments

No arguments

Return type

LoadNode object (or null if there are no more load nodes in the model).

Example

To get the load node in model m after load node ln:

```
var ln = ln.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load node.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load nodes from that model can be picked. If the argument is a Flag then only load nodes that are flagged with *limit* can be selected. If omitted, or null, any load nodes from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadNode object (or null if not picked)

Example

To pick a load node from model m giving the prompt 'Pick load node from screen':

```
var ln = LoadNode.Pick('Pick load node from screen', m);
```

Previous()

Description

Returns the previous load node in the model.

Arguments

No arguments

Return type

LoadNode object (or null if there are no more load nodes in the model).

Example

To get the load node in model m before load node ln:

```
var ln = ln.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load nodes using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load nodes
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load nodes from that model can be selected. If the argument is a Flag then only load nodes that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load nodes can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load nodes selected or null if menu cancelled

Example

To select load nodes from model m, flagging those selected with flag f, giving the prompt 'Select load nodes':

```
LoadNode.Select(f, 'Select load nodes', m);
```

To select load nodes, flagging those selected with flag f but limiting selection to load nodes flagged with flag l, giving the prompt 'Select load nodes':

```
LoadNode.Select(f, 'Select load nodes', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load node.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load node

Return type

No return value

Example

To set flag f for load node ln:

```
ln.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load node. The load node will be sketched until you either call LoadNode.Unsketch(), LoadNode.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load node is sketched. If omitted redraw is true. If you want to sketch several load nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load node ln:

```
ln.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load nodes in the model. The load nodes will be sketched until you either call LoadNode.Unsketch(), LoadNode.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load nodes will be sketched in
flag	Flag	Flag set on the load nodes that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load nodes are sketched. If omitted redraw is true. If you want to sketch flagged load nodes several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load nodes flagged with flag in model m:

```
LoadNode.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load nodes should be counted. If false or omitted referenced but undefined load nodes will also be included in the total.

Return type

number of load nodes

Example

To get the total number of load nodes in model m:

```
var total = LoadNode.Total(m);
```

Unblank()

Description

Unblanks the load node

Arguments

No arguments

Return type

No return value

Example

To unblank load node ln:

```
ln.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load nodes in model m:

```
LoadNode.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load nodes will be unblanked in
flag	Flag	Flag set on the load nodes that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load nodes in model m flagged with f:

```
LoadNode.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load nodes will be unset in
flag	Flag	Flag to unset on the load nodes

Return type

No return value

Example

To unset the flag f on all the load nodes in model m:

```
LoadNode.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load node.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load node is unsketched. If omitted redraw is true. If you want to unsketch several load nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load node ln:

```
ln.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load nodes.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load nodes in model m:

```
LoadNode.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load nodes will be unsketched in
flag	Flag	Flag set on the load nodes that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the load nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load nodes flagged with flag in model m:

```
LoadNode.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadNode object.

Example

To check if LoadNode property ln.example is a parameter by using the LoadNode.GetParameter() method:

```
if (ln.ViewParameters().GetParameter(ln.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load node ln:

```
ln.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load node.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load node ln:

```
var xrefs = ln.Xrefs();
```

---

toString()

Description

Creates a string containing the load node data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadNode.Keyword() and LoadNode.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for load node l in keyword format

```
var s = l.toString();
```

LoadRemovePart class

The LoadRemovePart class gives you access to define *LOAD_REMOVE_PART cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadRemovePart constants

Name	Description
LoadRemovePart.PART	LOAD is *LOAD_REMOVE_PART.

LoadRemovePart.SET_PART	LOAD is *LOAD_REMOVE_PART_SET.

LoadRemovePart properties

Name	Type	Description
exists	logical	true if LoadRemovePart exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the LoadRemovePart is in.
model	integer	The Model number that the load remove_part is in.
pid	integer	Part ID or Part Set ID
stgr	integer	Construction Stages ID at which part is removed.
time0	real	Time at which stress reduction starts.
time1	real	Time at which stresses become zero and elements are deleted.
type	constant	The Load RemovePart type. Can be LoadRemovePart.PART or LoadRemovePart.SET_PART.

Detailed Description

The LoadRemovePart class allows you to create, modify, edit and manipulate *LOAD_REMOVE_PART cards. See the documentation below for more details.

Constructor

new LoadRemovePart(Model*[Model]*, type*[constant]*, pid*[integer]*, time0 (optional)*[real]*, time1 (optional)*[real]*, stgr (optional)*[integer]*)

Description

Create a new LoadRemovePart object.

Arguments

Name	Type	Description
Model	Model	Model that LoadRemovePart will be created in
type	constant	Specify the type of LoadRemovePart (Can be LoadRemovePart.PART or LoadRemovePart.SET_PART
pid	integer	Part ID or Part Set ID
time0 (optional)	real	Time at which stress reduction starts.
time1 (optional)	real	Time at which stresses become zero and elements are deleted.
stgr (optional)	integer	Construction Stage ID at which part is removed.

Return type

LoadRemovePart object

Example

To create a new load remove_part in model m, of type PART, with pid 100, time0 2.5 and time1 4.5.

```
var l_r_p = new LoadRemovePart(m, LoadRemovePart.PART, 100, 2.5, 4.5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load remove_part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load remove_part

Return type

No return value

Example

To associate comment c to the load remove_part l_r_p:

```
l_r_p.AssociateComment(c);
```

Blank()

Description

Blanks the load remove_part

Arguments

No arguments

Return type

No return value

Example

To blank load remove_part l_r_p:

```
l_r_p.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load remove_parts in model m:

```
LoadRemovePart.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load remove_parts will be blanked in
flag	Flag	Flag set on the load remove_parts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load remove_parts in model m flagged with f:

```
LoadRemovePart.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load remove_part is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load remove_part l_r_p is blanked:

```
if (l_r_p.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load remove_part.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load remove_part

Return type

No return value

Example

To clear flag f for load remove_part l_r_p:

```
l_r_p.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load remove_part.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadRemovePart object

Example

To copy load remove_part l_r_p into load remove_part z:

```
var z = l_r_p.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load remove_part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load remove_part

Return type

No return value

Example

To detach comment c from the load remove_part l_r_p:

```
l_r_p.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load remove_part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load remove_part l_r_p:

```
l_r_p.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load remove_part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load remove_part in

Return type

LoadRemovePart object (or null if there are no load remove_parts in the model).

Example

To get the first load remove_part in model m:

```
var l_r_p = LoadRemovePart.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load remove_parts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts will be flagged in
flag	Flag	Flag to set on the load remove_parts

Return type

No return value

Example

To flag all of the load remove_parts with flag f in model m:

```
LoadRemovePart.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load remove_part is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load remove_part

Return type

true if flagged, false if not.

Example

To check if load remove_part l_r_p has flag f set on it:

```
if (l_r_p.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load remove_part in the model.

Note that ForEach has been designed to make looping over load remove_parts as fast as possible and so has some limitations.
Firstly, a single temporary LoadRemovePart object is created and on each function call it is updated with the current load remove_part data. This means that you should not try to store the LoadRemovePart object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load remove_parts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts are in
func	function	Function to call for each load remove_part
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load remove_parts in model m:

```
LoadRemovePart.ForEach(m, test);
function test(l_r_p)
{
// l_r_p is LoadRemovePart object
}
```

To call function test for all of the load remove_parts in model m with optional object:

```
var data = { x:0, y:0 };
LoadRemovePart.ForEach(m, test, data);
function test(l_r_p, extra)
{
// l_r_p is LoadRemovePart object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of LoadRemovePart objects for all of the load remove_parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load remove_parts from

Return type

Array of LoadRemovePart objects

Example

To make an array of LoadRemovePart objects for all of the load remove_parts in model m

```
var l_r_p = LoadRemovePart.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load remove_part.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load remove_part l_r_p:

```
var comm_array = l_r_p.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadRemovePart objects for all of the flagged load remove_parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load remove_parts from
flag	Flag	Flag set on the load remove_parts that you want to retrieve

Return type

Array of LoadRemovePart objects

Example

To make an array of LoadRemovePart objects for all of the load remove_parts in model m flagged with f

```
var l_r_p = LoadRemovePart.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadRemovePart object for a load remove_part ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load remove_part in
number	integer	number of the load remove_part you want the LoadRemovePart object for

Return type

LoadRemovePart object (or null if load remove_part does not exist).

Example

To get the LoadRemovePart object for load remove_part 100 in model m

```
var l_r_p = LoadRemovePart.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadRemovePart property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadRemovePart.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	load remove_part property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadRemovePart property l_r_p.example is a parameter:

```
Options.property_parameter_names = true;
if (l_r_p.GetParameter(l_r_p.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadRemovePart property l_r_p.example is a parameter by using the GetParameter method:

```
if (l_r_p.ViewParameters().GetParameter(l_r_p.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this LoadRemovePart (*LOAD_REMOVE_PART). **Note that a carriage return is not added**. See also LoadRemovePart.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for LoadRemovePart l_r_p:

```
var key = l_r_p.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the LoadRemovePart. **Note that a carriage return is not added**. See also LoadRemovePart.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for LoadRemovePart l_r_p:

```
var cards = l_r_p.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last load remove_part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load remove_part in

Return type

LoadRemovePart object (or null if there are no load remove_parts in the model).

Example

To get the last load remove_part in model m:

```
var l_r_p = LoadRemovePart.Last(m);
```

Next()

Description

Returns the next load remove_part in the model.

Arguments

No arguments

Return type

LoadRemovePart object (or null if there are no more load remove_parts in the model).

Example

To get the load remove_part in model m after load remove_part l_r_p:

```
var l_r_p = l_r_p.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load remove_part.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load remove_parts from that model can be picked. If the argument is a Flag then only load remove_parts that are flagged with *limit* can be selected. If omitted, or null, any load remove_parts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.

button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadRemovePart object (or null if not picked)

Example

To pick a load remove_part from model m giving the prompt 'Pick load remove_part from screen':

```
var l_r_p = LoadRemovePart.Pick('Pick load remove_part from screen', m);
```

Previous()

Description

Returns the previous load remove_part in the model.

Arguments

No arguments

Return type

LoadRemovePart object (or null if there are no more load remove_parts in the model).

Example

To get the load remove_part in model m before load remove_part l_r_p:

```
var l_r_p = l_r_p.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load remove_parts using standard PRIMER object menus.

Arguments

Name	**Type**	**Description**
flag	Flag	Flag to use when selecting load remove_parts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load remove_parts from that model can be selected. If the argument is a Flag then only load remove_parts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load remove_parts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load remove_parts selected or null if menu cancelled

Example

To select load remove_parts from model m, flagging those selected with flag f, giving the prompt 'Select load remove_parts':

```
LoadRemovePart.Select(f, 'Select load remove_parts', m);
```

To select load remove_parts, flagging those selected with flag f but limiting selection to load remove_parts flagged with flag l, giving the prompt 'Select load remove_parts':

```
LoadRemovePart.Select(f, 'Select load remove_parts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load remove_part.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load remove_part

Return type

No return value

Example

To set flag f for load remove_part l_r_p:

```
l_r_p.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load remove_part. The load remove_part will be sketched until you either call LoadRemovePart.Unsketch(), LoadRemovePart.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load remove_part is sketched. If omitted redraw is true. If you want to sketch several load remove_parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load remove_part l_r_p:

```
l_r_p.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load remove_parts in the model. The load remove_parts will be sketched until you either call LoadRemovePart.Unsketch(), LoadRemovePart.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load remove_parts will be sketched in
flag	Flag	Flag set on the load remove_parts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load remove_parts are sketched. If omitted redraw is true. If you want to sketch flagged load remove_parts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load remove_parts flagged with flag in model m:

```
LoadRemovePart.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load remove_parts should be counted. If false or omitted referenced but undefined load remove_parts will also be included in the total.

Return type

number of load remove_parts

Example

To get the total number of load remove_parts in model m:

```
var total = LoadRemovePart.Total(m);
```

Unblank()

Description

Unblanks the load remove_part

Arguments

No arguments

Return type

No return value

Example

To unblank load remove_part l_r_p:

```
l_r_p.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load remove_parts in model m:

```
LoadRemovePart.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load remove_parts will be unblanked in
flag	Flag	Flag set on the load remove_parts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load remove_parts in model m flagged with f:

```
LoadRemovePart.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load remove_parts will be unset in
flag	Flag	Flag to unset on the load remove_parts

Return type

No return value

Example

To unset the flag f on all the load remove_parts in model m:

```
LoadRemovePart.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load remove_part.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load remove_part is unsketched. If omitted redraw is true. If you want to unsketch several load remove_parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load remove_part l_r_p:

```
l_r_p.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load remove_parts.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load remove_parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load remove_parts in model m:

```
LoadRemovePart.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load remove_parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load remove_parts will be unsketched in
flag	Flag	Flag set on the load remove_parts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the load remove_parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load remove_parts flagged with flag in model m:

```
LoadRemovePart.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadRemovePart object.

Example

To check if LoadRemovePart property l_r_p.example is a parameter by using the LoadRemovePart.GetParameter() method:

```
if (l_r_p.ViewParameters().GetParameter(l_r_p.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load remove_part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load remove_part l_r_p:

```
l_r_p.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load remove_part.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load remove_part l_r_p:

```
var xrefs = l_r_p.Xrefs();
```

---

toString()

Description

Creates a string containing the LoadRemovePart data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadRemovePart.Keyword() and LoadRemovePart.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for LoadRemovePart l_r_p in keyword format

```
var s = l_r_p.toString();
```

LoadRigidBody class

The LoadRigidBody class gives you access to define load rigidbody cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadRigidBody properties

Name	Type	Description
cid	integer	Coordinate system ID

dof	integer	Applicable degrees-of-freedom
exists	logical	true if load rigidbody exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the load rigidbody is in.
lcid	integer	Curve ID
m1	integer	Node 1 ID
m2	integer	Node 2 ID
m3	integer	Node 3 ID
model	integer	The Model number that the load rigidbody is in.
pid	integer	Part ID
sf	real	Curve scale factor

Detailed Description

The LoadRigidBody class allows you to create, modify, edit and manipulate load rigidbody cards. See the documentation below for more details.

Constructor

new LoadRigidBody(Model*[Model]*, pid*[integer]*, dof*[integer]*, lcid*[integer]*, sf (optional)*[real]*, cid (optional)*[integer]*, m1 (optional)*[integer]*, m2 (optional)*[integer]*, m3 (optional)*[integer]*)

Description

Create a new LoadRigidBody object.

Arguments

Name	Type	Description
Model	Model	Model that load rigidbody will be created in
pid	integer	Part ID
dof	integer	Applicable degrees-of-freedom
lcid	integer	Curve ID
sf (optional)	real	Curve scale factor
cid (optional)	integer	Coordinate system ID
m1 (optional)	integer	Node 1 ID
m2 (optional)	integer	Node 2 ID
m3 (optional)	integer	Node 3 ID

Return type

LoadRigidBody object

Example

To create a new load rigidbody in model m, for part 100, with loadcurve 9 and a scale factor of 0.5

```
var lrb = new LoadRigidBody(m, 100, 2, 9, 0.5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load rigidbody.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load rigidbody

Return type

No return value

Example

To associate comment c to the load rigidbody lrb:

```
lrb.AssociateComment(c);
```

Blank()

Description

Blanks the load rigidbody

Arguments

No arguments

Return type

No return value

Example

To blank load rigidbody lrb:

```
lrb.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load rigidbodies in model m:

```
LoadRigidBody.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load rigidbodies will be blanked in
flag	Flag	Flag set on the load rigidbodies that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load rigidbodies in model m flagged with f:

```
LoadRigidBody.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load rigidbody is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load rigidbody lrb is blanked:

```
if (lrb.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load rigidbody.

Arguments

Name	Type	Description

flag	Flag	Flag to clear on the load rigidbody

Return type

No return value

Example

To clear flag f for load rigidbody lrb:

```
lrb.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load rigidbody.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadRigidBody object

Example

To copy load rigidbody lrb into load rigidbody z:

```
var z = lrb.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load rigidbody.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load rigidbody

Return type

No return value

Example

To detach comment c from the load rigidbody lrb:

```
lrb.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load rigidbody. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load rigidbody lrb:

```
lrb.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load rigidbody in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load rigidbody in

Return type

LoadRigidBody object (or null if there are no load rigidbodies in the model).

Example

To get the first load rigidbody in model m:

```
var lrb = LoadRigidBody.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load rigidbodies in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies will be flagged in
flag	Flag	Flag to set on the load rigidbodies

Return type

No return value

Example

To flag all of the load rigidbodies with flag f in model m:

```
LoadRigidBody.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load rigidbody is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load rigidbody

Return type

true if flagged, false if not.

Example

To check if load rigidbody lrb has flag f set on it:

```
if (lrb.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load rigidbody in the model.
Note that ForEach has been designed to make looping over load rigidbodies as fast as possible and so has some limitations.
Firstly, a single temporary LoadRigidBody object is created and on each function call it is updated with the current load rigidbody data. This means that you should not try to store the LoadRigidBody object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load rigidbodies inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies are in
func	function	Function to call for each load rigidbody
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load rigidbodies in model m:

```
LoadRigidBody.ForEach(m, test);
function test(lrb)
{
// lrb is LoadRigidBody object
}
```

To call function test for all of the load rigidbodies in model m with optional object:

```
var data = { x:0, y:0 };
LoadRigidBody.ForEach(m, test, data);
function test(lrb, extra)
{
// lrb is LoadRigidBody object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of LoadRigidBody objects for all of the load rigidbodies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load rigidbodies from

Return type

Array of LoadRigidBody objects

Example

To make an array of LoadRigidBody objects for all of the load rigidbodies in model m

```
var lrb = LoadRigidBody.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load rigidbody.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load rigidbody lrb:

```
var comm_array = lrb.GetComments();
```

GetFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Returns an array of LoadRigidBody objects for all of the flagged load rigidbodies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load rigidbodies from
flag	Flag	Flag set on the load rigidbodies that you want to retrieve

Return type

Array of LoadRigidBody objects

Example

To make an array of LoadRigidBody objects for all of the load rigidbodies in model m flagged with f

```
var lrb = LoadRigidBody.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadRigidBody object for a load rigidbody ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load rigidbody in
number	integer	number of the load rigidbody you want the LoadRigidBody object for

Return type

LoadRigidBody object (or null if load rigidbody does not exist).

Example

To get the LoadRigidBody object for load rigidbody 100 in model m

```
var lrb = LoadRigidBody.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadRigidBody property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadRigidBody.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load rigidbody property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadRigidBody property lrb.example is a parameter:

```
Options.property_parameter_names = true;
if (lrb.GetParameter(lrb.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadRigidBody property lrb.example is a parameter by using the GetParameter method:

```
if (lrb.ViewParameters().GetParameter(lrb.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this load rigidbody (*LOAD_RIGIDBODY). **Note that a carriage return is not added**. See also LoadRigidBody.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for load rigidbody lrb:

```
var key = lrb.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the load rigidbody. **Note that a carriage return is not added**. See also LoadRigidBody.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for load rigidbody lrb:

```
var cards = lrb.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last load rigidbody in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load rigidbody in

Return type

LoadRigidBody object (or null if there are no load rigidbodies in the model).

Example

To get the last load rigidbody in model m:

```
var lrb = LoadRigidBody.Last(m);
```

Next()

Description

Returns the next load rigidbody in the model.

Arguments

No arguments

Return type

LoadRigidBody object (or null if there are no more load rigidbodies in the model).

Example

To get the load rigidbody in model m after load rigidbody lrb:

```
var lrb = lrb.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load rigidbody.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load rigidbodies from that model can be picked. If the argument is a Flag then only load rigidbodies that are flagged with *limit* can be selected. If omitted, or null, any load rigidbodies from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadRigidBody object (or null if not picked)

Example

To pick a load rigidbody from model m giving the prompt 'Pick load rigidbody from screen':

```
var lrb = LoadRigidBody.Pick('Pick load rigidbody from screen', m);
```

Previous()

Description

Returns the previous load rigidbody in the model.

Arguments

No arguments

Return type

LoadRigidBody object (or null if there are no more load rigidbodies in the model).

Example

To get the load rigidbody in model m before load rigidbody lrb:

```
var lrb = lrb.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load rigidbodies using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load rigidbodies
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load rigidbodies from that model can be selected. If the argument is a Flag then only load rigidbodies that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load rigidbodies can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load rigidbodies selected or null if menu cancelled

Example

To select load rigidbodies from model m, flagging those selected with flag f, giving the prompt 'Select load rigidbodies':

```
LoadRigidBody.Select(f, 'Select load rigidbodies', m);
```

To select load rigidbodies, flagging those selected with flag f but limiting selection to load rigidbodies flagged with flag l, giving the prompt 'Select load rigidbodies':

```
LoadRigidBody.Select(f, 'Select load rigidbodies', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load rigidbody.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load rigidbody

Return type

No return value

Example

To set flag f for load rigidbody lrb:

```
lrb.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load rigidbody. The load rigidbody will be sketched until you either call LoadRigidBody.Unsketch(), LoadRigidBody.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load rigidbody is sketched. If omitted redraw is true. If you want to sketch several load rigidbodies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load rigidbody lrb:

```
lrb.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load rigidbodies in the model. The load rigidbodies will be sketched until you either call LoadRigidBody.Unsketch(), LoadRigidBody.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load rigidbodies will be sketched in
flag	Flag	Flag set on the load rigidbodies that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load rigidbodies are sketched. If omitted redraw is true. If you want to sketch flagged load rigidbodies several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load rigidbodies flagged with flag in model m:

```
LoadRigidBody.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load rigidbodies should be counted. If false or omitted referenced but undefined load rigidbodies will also be included in the total.

Return type

number of load rigidbodies

Example

To get the total number of load rigidbodies in model m:

```
var total = LoadRigidBody.Total(m);
```

Unblank()

Description

Unblanks the load rigidbody

Arguments

No arguments

Return type

No return value

Example

To unblank load rigidbody lrb:

```
lrb.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load rigidbodies in model m:

```
LoadRigidBody.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load rigidbodies will be unblanked in

flag	Flag	Flag set on the load rigidbodies that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load rigidbodies in model m flagged with f:

```
LoadRigidBody.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load rigidbodies will be unset in
flag	Flag	Flag to unset on the load rigidbodies

Return type

No return value

Example

To unset the flag f on all the load rigidbodies in model m:

```
LoadRigidBody.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load rigidbody.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load rigidbody is unsketched. If omitted redraw is true. If you want to unsketch several load rigidbodies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load rigidbody lrb:

```
lrb.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load rigidbodies.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load rigidbodies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load rigidbodies in model m:

```
LoadRigidBody.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load rigidbodies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load rigidbodies will be unsketched in
flag	Flag	Flag set on the load rigidbodies that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the load rigidbodies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load rigidbodies flagged with flag in model m:

```
LoadRigidBody.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadRigidBody object.

Example

To check if LoadRigidBody property lrb.example is a parameter by using the LoadRigidBody.GetParameter() method:

```
if (lrb.ViewParameters().GetParameter(lrb.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load rigidbody. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load rigidbody lrb:

```
lrb.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load rigidbody.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load rigidbody lrb:

```
var xrefs = lrb.Xrefs();
```

---

toString()

Description

Creates a string containing the load rigidbody data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadRigidBody.Keyword() and LoadRigidBody.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for load rigidbody lrb in keyword format

```
var s = lrb.toString();
```

LoadShell class

The LoadShell class gives you access to define *LOAD_SHELL cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

LoadShell constants

Name	Description
LoadShell.ELEMENT	Load is *LOAD_SHELL_ELEMENT.
LoadShell.SET	LOAD is *LOAD_SHELL_SET.

LoadShell properties

Name	Type	Description
at	real	Arrival time for pressure
eid	integer	Shell ID or shell set ID
exists	logical	true if LoadShell exists, false if referred to but not defined. (read only)
heading	string	LoadShell heading
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the LoadShell is in.
label	integer	LoadShell number.
lcid	integer	Curve ID
lsid	integer	LoadShell number (identical to label).
model	integer	The Model number that the load shell is in.
sf	real	Curve scale factor
type	constant	The Load Node type. Can be LoadShell.ELEMENT or LoadShell.SET.

Detailed Description

The LoadShell class allows you to create, modify, edit and manipulate *LOAD_SHELL cards. See the documentation below for more details.

Constructor

new LoadShell(Model*[Model]*, type*[constant]*, eid*[integer]*, lcid*[integer]*, sf (optional)*[real]*, at (optional)*[real]*, lsid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new LoadShell object.

Arguments

Name	Type	Description
Model	Model	Model that LoadShell will be created in
type	constant	Specify the type of LoadShell (Can be LoadShell.ELEMENT or LoadShell.SET)
eid	integer	Shell ID or shell set ID
lcid	integer	Curve ID
sf (optional)	real	Curve scale factor
at (optional)	real	Arrival time for pressure
lsid (optional)	integer	LoadShell number
heading (optional)	string	Title for the LoadShell

Return type

LoadShell object

Example

To create a new load shell in model m, of type SET, with loadcurve 9 and a scale factor of 0.5

```
var b = new LoadShell(m, LoadShell.SET, 100, 2, 9, 0.5);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a load shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the load shell

Return type

No return value

Example

To associate comment c to the load shell ls:

```
ls.AssociateComment(c);
```

Blank()

Description

Blanks the load shell

Arguments

No arguments

Return type

No return value

Example

To blank load shell ls:

```
ls.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load shells will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load shells in model m:

```
LoadShell.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load shells will be blanked in
flag	Flag	Flag set on the load shells that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the load shells in model m flagged with f:

```
LoadShell.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the load shell is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if load shell ls is blanked:

```
if (ls.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the load shell.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the load shell

Return type

No return value

Example

To clear flag f for load shell ls:

```
ls.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the load shell.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

LoadShell object

Example

To copy load shell ls into load shell z:

```
var z = ls.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a load shell.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the load shell

Return type

No return value

Example

To detach comment c from the load shell ls:

```
ls.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for load shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for load shell ls:

```
ls.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first load shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first load shell in

Return type

LoadShell object (or null if there are no load shells in the model).

Example

To get the first load shell in model m:

```
var ls = LoadShell.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free load shell label in the model. Also see LoadShell.LastFreeLabel(), LoadShell.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free load shell label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

LoadShell label.

Example

To get the first free load shell label in model m:

```
var label = LoadShell.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the load shells in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all load shells will be flagged in
flag	Flag	Flag to set on the load shells

Return type

No return value

Example

To flag all of the load shells with flag f in model m:

```
LoadShell.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the load shell is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the load shell

Return type

true if flagged, false if not.

Example

To check if load shell ls has flag f set on it:

```
if (ls.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each load shell in the model.
Note that ForEach has been designed to make looping over load shells as fast as possible and so has some limitations.
Firstly, a single temporary LoadShell object is created and on each function call it is updated with the current load shell data. This means that you should not try to store the LoadShell object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new load shells inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all load shells are in
func	function	Function to call for each load shell
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the load shells in model m:

```
LoadShell.ForEach(m, test);
function test(ls)
{
// ls is LoadShell object
}
```

To call function test for all of the load shells in model m with optional object:

```
var data = { x:0, y:0 };
LoadShell.ForEach(m, test, data);
function test(ls, extra)
{
// ls is LoadShell object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of LoadShell objects for all of the load shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load shells from

Return type

Array of LoadShell objects

Example

To make an array of LoadShell objects for all of the load shells in model m

```
var ls = LoadShell.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a load shell.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the load shell ls:

```
var comm_array = ls.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of LoadShell objects for all of the flagged load shells in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get load shells from
flag	Flag	Flag set on the load shells that you want to retrieve

Return type

Array of LoadShell objects

Example

To make an array of LoadShell objects for all of the load shells in model m flagged with f

```
var ls = LoadShell.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the LoadShell object for a load shell ID.

Arguments

Name	Type	Description
Model	Model	Model to find the load shell in
number	integer	number of the load shell you want the LoadShell object for

Return type

LoadShell object (or null if load shell does not exist).

Example

To get the LoadShell object for load shell 100 in model m

```
var ls = LoadShell.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a LoadShell property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the LoadShell.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	load shell property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if LoadShell property ls.example is a parameter:

```
Options.property_parameter_names = true;
if (ls.GetParameter(ls.example) ) do_something...
Options.property_parameter_names = false;
```

To check if LoadShell property ls.example is a parameter by using the GetParameter method:

```
if (ls.ViewParameters().GetParameter(ls.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this LoadShell (*LOAD_SHELL_xxxx). **Note that a carriage return is not added**. See also LoadShell.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for LoadShell m:

```
var key = m.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the LoadShell. **Note that a carriage return is not added**. See also LoadShell.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for LoadShell l:

```
var cards = l.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last load shell in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last load shell in

Return type

LoadShell object (or null if there are no load shells in the model).

Example

To get the last load shell in model m:

```
var ls = LoadShell.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free load shell label in the model. Also see LoadShell.FirstFreeLabel(), LoadShell.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free load shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

LoadShell label.

Example

To get the last free load shell label in model m:

```
var label = LoadShell.LastFreeLabel(m);
```

Next()

Description

Returns the next load shell in the model.

Arguments

No arguments

Return type

LoadShell object (or null if there are no more load shells in the model).

Example

To get the load shell in model m after load shell ls:

```
var ls = ls.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) load shell label in the model. Also see LoadShell.FirstFreeLabel(), LoadShell.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free load shell label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

LoadShell label.

Example

To get the next free load shell label in model m:

```
var label = LoadShell.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a load shell.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load shells from that model can be picked. If the argument is a Flag then only load shells that are flagged with *limit* can be selected. If omitted, or null, any load shells from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

LoadShell object (or null if not picked)

Example

To pick a load shell from model m giving the prompt 'Pick load shell from screen':

```
var ls = LoadShell.Pick('Pick load shell from screen', m);
```

Previous()

Description

Returns the previous load shell in the model.

Arguments

No arguments

Return type

LoadShell object (or null if there are no more load shells in the model).

Example

To get the load shell in model m before load shell ls:

```
var ls = ls.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load shells will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the load shells in model m, from 1000000:

```
LoadShell.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load shells will be renumbered in
flag	Flag	Flag set on the load shells that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the load shells in model m flagged with f, from 1000000:

```
LoadShell.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select load shells using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting load shells
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only load shells from that model can be selected. If the argument is a Flag then only load shells that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any load shells can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of load shells selected or null if menu cancelled

Example

To select load shells from model m, flagging those selected with flag f, giving the prompt 'Select load shells':

```
LoadShell.Select(f, 'Select load shells', m);
```

To select load shells, flagging those selected with flag f but limiting selection to load shells flagged with flag l, giving the prompt 'Select load shells':

```
LoadShell.Select(f, 'Select load shells', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the load shell.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the load shell

Return type

No return value

Example

To set flag f for load shell ls:

```
ls.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the load shell. The load shell will be sketched until you either call LoadShell.Unsketch(), LoadShell.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load shell is sketched. If omitted redraw is true. If you want to sketch several load shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch load shell ls:

```
ls.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged load shells in the model. The load shells will be sketched until you either call LoadShell.Unsketch(), LoadShell.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged load shells will be sketched in
flag	Flag	Flag set on the load shells that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the load shells are sketched. If omitted redraw is true. If you want to sketch flagged load shells several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all load shells flagged with flag in model m:

```
LoadShell.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing load shells should be counted. If false or omitted referenced but undefined load shells will also be included in the total.

Return type

number of load shells

Example

To get the total number of load shells in model m:

```
var total = LoadShell.Total(m);
```

Unblank()

Description

Unblanks the load shell

Arguments

No arguments

Return type

No return value

Example

To unblank load shell ls:

```
ls.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the load shells in the model.

Arguments

Name	Type	Description

Model	Model	Model that all load shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load shells in model m:

```
LoadShell.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged load shells will be unblanked in
flag	Flag	Flag set on the load shells that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the load shells in model m flagged with f:

```
LoadShell.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all load shells will be unset in
flag	Flag	Flag to unset on the load shells

Return type

No return value

Example

To unset the flag f on all the load shells in model m:

```
LoadShell.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the load shell.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the load shell is unsketched. If omitted redraw is true. If you want to unsketch several load shells and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch load shell ls:

```
ls.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all load shells.

Arguments

Name	Type	Description
Model	Model	Model that all load shells will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the load shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load shells in model m:

```
LoadShell.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged load shells in the model.

Arguments

Name	Type	Description
Model	Model	Model that all load shells will be unsketched in
flag	Flag	Flag set on the load shells that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the load shells are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all load shells flagged with flag in model m:

```
LoadShell.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

LoadShell object.

Example

To check if LoadShell property ls.example is a parameter by using the LoadShell.GetParameter() method:

```
if (ls.ViewParameters().GetParameter(ls.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for load shell. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for load shell ls:

```
ls.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this load shell.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for load shell ls:

```
var xrefs = ls.Xrefs();
```

---

toString()

Description

Creates a string containing the LoadShell data in keyword format. Note that this contains the keyword header and the keyword cards. See also LoadShell.Keyword() and LoadShell.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for LoadShell l in keyword format

```
var s = l.toString();
```

Material class

The Material class gives you access to material cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AddOptionalCards()
- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- Density()
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetErosionPropertyByName(acronym*[string]*, idam_index (optional)*[integer]*)
- GetMaterialErosionExists()
- GetParameter(prop*[string]*)
- GetPropertyByIndex(index*[integer]*)
- GetPropertyByName(acronym*[string]*)
- GetPropertyByRowCol(row*[integer]*, col*[integer]*)
- GetPropertyNameForIndex(index*[integer]*)
- GetPropertyNameForRowCol(row*[integer]*, col*[integer]*)
- Keyword(index (optional)*[integer]*)
- KeywordCards(index (optional)*[integer]*)

- Next()
- PoissonsRatio()
- Previous()
- RemoveMaterialErosion()
- SetErosionPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*, idam_index (optional)*[integer]*)
- SetFlag(flag*[Flag]*)
- SetMaterialErosion()
- SetPropertyByIndex(index*[integer]*, value*[integer/real for numeric properties, string for character properties]*)
- SetPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*)
- SetPropertyByRowCol(row*[integer]*, col*[integer]*, value*[integer/real for numeric properties, string for character properties]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- YieldStress()
- YoungsModulus()
- toString()

Material properties

Name	Type	Description
addKeywords	integer	The number of **extra** *MAT_ADD_xxxx keywords that this material definition has (read only). Note that if there is only a single *MAT_ADD_xxxx keyword for an ID this will be 0. For example, if for material 1 both a *MAT_PIECEWISE_LINEAR_PLASTICITY card and a *MAT_ADD_EROSION card exist then this will return 1. If for material 2 only a *MAT_ADD_EROSION card exists then this will return 0. Also see Material.Keyword() and Material.KeywordCards()
colour	Colour	The colour of the material
cols	real	The number of columns of data the material has (read only)
exists	logical	true if material exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the material is in.
label	integer or string	Material number or character label. Also see the mid property which is an alternative name for this.
mid	integer or string	Material number or character label. Also see the label property which is an alternative name for this.
model	integer	The Model number that the material is in.
optionalCards	integer	The number of optional extra cards that this material definition can have (read only). Also see Material.AddOptionalCards()
properties	integer	The total number of properties that the material has
rows	integer	The number of rows of data the material has (read only)
title	string	Material title
transparency	integer	The transparency of the material (0-100) 0% is opaque, 100% is transparent.
type	string	The material type (e.g. 'ELASTIC', 'RIGID' etc).

Detailed Description

The Material class allows you to create, modify, edit and manipulate material cards. See the documentation below for more details.

Constructor

new Material(Model*[Model]*, mid*[integer or string]*, type*[string]*)

Description

Create a new Material object.

Arguments

Name	Type	Description
Model	Model	Model that material will be created in
mid	integer or string	Material number or character label
type	string	Material type. Either give the LS-DYNA material name or 3 digit number.

Return type

Material object

Example

To create a new rigid material in model m with label 100

```
var mat = new Material(m, 100, "RIGID");
```

or

```
var mat = new Material(m, 100, "020");
```

or

```
var mat = new Material(m, 100, "*MAT_RIGID");
```

or

```
var mat = new Material(m, 100, "*MAT_020");
```

Details of functions

AddOptionalCards()

Description

Adds any optional cards for the material.
Some materials have extra optional cards in the input. If they are there LS-DYNA will read them but they are not required input. For example a material could have three required cards and one extra optional card. If PRIMER reads this material from a keyword file and it only has the three required cards then the properties in the material will only be defined for those cards. i.e. there will not be any properties in the material for the extra optional line.
If you edit the material interactively in PRIMER then the extra optional card will be shown so you can add values if required. When writing the material to a keyword file the extra optional card will be omitted if none of the fields are used.
If you want to add one of the properties for the extra optional card in JavaScript this method will ensure that the extra card is defined and the properties added to the material as zero values. You can then use Material.SetPropertyByIndex(), Material.SetPropertyByName() or Material.SetPropertyByRowCol() as normal to set the properties. Also see the optionalCards property.

Arguments

No arguments

Return type

no return value

Example

To add any optional cards for material m:

```
m.AddOptionalCards();
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a material.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the material

Return type

No return value

Example

To associate comment c to the material m:

```
m.AssociateComment(c);
```

Blank()

Description

Blanks the material

Arguments

No arguments

Return type

No return value

Example

To blank material m:

```
m.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that all materials will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the materials in model m:

```
Material.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged materials will be blanked in
flag	Flag	Flag set on the materials that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the materials in model m flagged with f:

```
Material.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the material is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if material m is blanked:

```
if (m.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse material m:

```
m.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the material.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the material

Return type

No return value

Example

To clear flag f for material m:

```
m.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the material.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Material object

Example

To copy material m into material z:

```
var z = m.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a material.

Arguments

Name	Type	Description
Model	Model	Model that the material will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Material object (or null if not made)

Example

To start creating a material in model m:

```
var mat = Material.Create(m);
```

Density()

Description

Get the density material.

Arguments

No arguments

Return type

real

Example

To get the density for material m:

```
var density = m.Density();
```

DetachComment(Comment[*Comment*])

Description

Detaches a comment from a material.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the material

Return type

No return value

Example

To detach comment c from the material m:

```
m.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit material m:

```
m.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for material. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for material m:

```
m.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for material.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the material colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the material.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing material m:

```
var colour = m.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first material in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first material in

Return type

Material object (or null if there are no materials in the model).

Example

To get the first material in model m:

```
var m = Material.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free material label in the model. Also see Material.LastFreeLabel(), Material.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free material label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Material label.

Example

To get the first free material label in model m:

```
var label = Material.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the materials in the model with a defined flag.

Arguments

Name	Type	Description

Model	Model	Model that all materials will be flagged in
flag	Flag	Flag to set on the materials

Return type

No return value

Example

To flag all of the materials with flag f in model m:

```
Material.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the material is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the material

Return type

true if flagged, false if not.

Example

To check if material m has flag f set on it:

```
if (m.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each material in the model.
Note that ForEach has been designed to make looping over materials as fast as possible and so has some limitations.
Firstly, a single temporary Material object is created and on each function call it is updated with the current material data. This means that you should not try to store the Material object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new materials inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all materials are in
func	function	Function to call for each material
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the materials in model m:

```
Material.ForEach(m, test);
function test(m)
{
// m is Material object
}
```

To call function test for all of the materials in model m with optional object:

```
var data = { x:0, y:0 };
Material.ForEach(m, test, data);
function test(m, extra)
{
// m is Material object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Material objects for all of the materials in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get materials from

Return type

Array of Material objects

Example

To make an array of Material objects for all of the materials in model m

```
var m = Material.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a material.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the material m:

```
var comm_array = m.GetComments();
```

GetErosionPropertyByName(acronym*[string]*, idam_index (optional)*[integer]*)

Description

Returns the value of Erosion property string *acronym* for this Material object or null if Erosion is not set on Material or no such Erosion property exists.

Arguments

Name	Type	Description
acronym	string	The acronym of the Erosion property value to retrieve
idam_index (optional)	integer	Required if property is one of IDAM cards pair property (for IDAM value less than zero) . If the argument is not given, returns the property value for first IDAM cards Pair. The index value starts from zero.

Return type

Property value (real/integer)

Example

To return the value of IDAM for material m:

```
var idam = m.GetErosionPropertyByName("IDAM");
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Material objects for all of the flagged materials in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get materials from
flag	Flag	Flag set on the materials that you want to retrieve

Return type

Array of Material objects

Example

To make an array of Material objects for all of the materials in model m flagged with f

```
var m = Material.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Material object for a material ID.

Arguments

Name	Type	Description
Model	Model	Model to find the material in
number	integer	number of the material you want the Material object for

Return type

Material object (or null if material does not exist).

Example

To get the Material object for material 100 in model m

```
var m = Material.GetFromID(m, 100);
```

GetMaterialErosionExists()

Description

Checks if the Erosion properties are defined for this Material object.

Arguments

No arguments

Return type

logical

Example

To get whether the Material has Erosion Properties:

```
m.GetMaterialErosionExists();
```

GetParameter(prop*[string]*)

Description

Checks if a Material property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Material.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	material property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Material property m.example is a parameter:

```
Options.property_parameter_names = true;
if (m.GetParameter(m.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Material property m.example is a parameter by using the GetParameter method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

GetPropertyByIndex(index*[integer]*)

Description

Returns the value of property at index *index* for this Material object or null if no property exists.

Arguments

Name	Type	Description
index	integer	The index of the property value to retrieve. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for a material may not have adjacent indices.

Return type

Property value (real/integer)

Example

To return the property at index 2, for material m:

```
var prop = m.GetPropertyByIndex(2);
```

GetPropertyByName(acronym*[string]*)

Description

Returns the value of property string *acronym* for this Material object or null if no property exists.

Arguments

Name	Type	Description
acronym	string	The acronym of the property value to retrieve

Return type

Property value (real/integer)

Example

To return the value of RO for material m:

```
var ro = m.GetPropertyByName("RO");
```

GetPropertyByRowCol(row*[integer]*, col*[integer]*)

Description

Returns the value of the property for row and col for this Material object or null if no property exists. **Note that rows and columns start at 0**.

Arguments

Name	Type	Description
row	integer	The row of the property value to retrieve
col	integer	The column of the property value to retrieve

Return type

Property value (real/integer)

Example

To return the value of the property at row 0, column 1 for material m:

```
var prop = m.GetPropertyByRowCol(0, 1);
```

GetPropertyNameForIndex(index*[integer]*)

Description

Returns the name of the property at index *index* for this Material object or null if there is no property.

Arguments

Name	Type	Description
index	integer	The index of the property name to retrieve. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for a material may not have adjacent indices.

Return type

Property name (string)

Example

To return the name of the property at index 2, for material m:

```
var name = m.GetPropertyNameForIndex(2);
```

GetPropertyNameForRowCol(row*[integer]*, col*[integer]*)

Description

Returns the name of the property at row and col for this Material object or null if there is no property. **Note that rows and columns start at 0**.

Arguments

Name	Type	Description
row	integer	The row of the property name to retrieve
col	integer	The column of the property name to retrieve

Return type

Property name (string)

Example

To return the name of the property at row 0, column 1 for material m:

```
var name = m.GetPropertyNameForRowCol(0, 1);
```

Keyword(index (optional)*[integer]*)

Description

Returns the keyword for this material (e.g. *MAT_RIGID, *MAT_ELASTIC etc). **Note that a carriage return is not added**. See also Material.KeywordCards()

Arguments

Name	Type	Description
index (optional)	integer	If this argument is not given then the material keyword is returned as normal. However if the material also has *MAT_ADD_xxxx cards defined for it (e.g. *MAT_ADD_EROSION) then the index can be used to return the title for the *MAT_ADD card instead. The index value starts from zero. The number of *MAT_ADD cards can be found from the addKeywords property

Return type

string containing the keyword.

Example

To get the keyword for material m:

```
var key = m.Keyword();
```

To print all of the keywords and keyword cards for any *MAT_ADD cards for material m:

```
for (i=0; i<m.addKeywords; i++)
{
    Message(m.Keyword(i));
    Message(m.KeywordCards(i));
}
```

KeywordCards(index (optional)*[integer]*)

Description

Returns the keyword cards for the material. **Note that a carriage return is not added**. See also Material.Keyword()

Arguments

Name	Type	Description
index (optional)	integer	If this argument is not given then the material keyword cards are returned as normal. However if the material also has *MAT_ADD_xxxx cards defined for it (e.g. *MAT_ADD_EROSION) then the index can be used to return the cards for the *MAT_ADD card instead. The index value starts from zero. The number of *MAT_ADD cards can be found from the addKeywords property

Return type

string containing the cards.

Example

To get the cards for material m:

```
var cards = m.KeywordCards();
```

To print all of the keywords and keyword cards for any *MAT_ADD cards for material m:

```
for (i=0; i<m.addKeywords; i++)
{
    Message(m.Keyword(i));
    Message(m.KeywordCards(i));
}
```

Last(Model*[Model]*) [static]

Description

Returns the last material in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last material in

Return type

Material object (or null if there are no materials in the model).

Example

To get the last material in model m:

```
var m = Material.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free material label in the model. Also see Material.FirstFreeLabel(), Material.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free material label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Material label.

Example

To get the last free material label in model m:

```
var label = Material.LastFreeLabel(m);
```

Next()

Description

Returns the next material in the model.

Arguments

No arguments

Return type

Material object (or null if there are no more materials in the model).

Example

To get the material in model m after material m:

```
var m = m.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) material label in the model. Also see Material.FirstFreeLabel(), Material.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free material label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Material label.

Example

To get the next free material label in model m:

```
var label = Material.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a material.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only materials from that model can be picked. If the argument is a Flag then only materials that are flagged with *limit* can be selected. If omitted, or null, any materials from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Material object (or null if not picked)

Example

To pick a material from model m giving the prompt 'Pick material from screen':

```
var m = Material.Pick('Pick material from screen', m);
```

PoissonsRatio()

Description

Get Poissons ratio for the material.

Arguments

No arguments

Return type

real

Example

To get Poissons ratio for material m:

```
var pr = m.PoissonsRatio(f);
```

Previous()

Description

Returns the previous material in the model.

Arguments

No arguments

Return type

Material object (or null if there are no more materials in the model).

Example

To get the material in model m before material m:

```
var m = m.Previous();
```

RemoveMaterialErosion()

Description

Removes the Erosion properties for this Material object.

Arguments

No arguments

Return type

No return value

Example

To remove the Erosion properties for material m:

```
m.RemoveMaterialErosion();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the materials in the model.

Arguments

Name	Type	Description

Model	Model	Model that all materials will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the materials in model m, from 1000000:

```
Material.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged materials will be renumbered in
flag	Flag	Flag set on the materials that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the materials in model m flagged with f, from 1000000:

```
Material.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select materials using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting materials
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only materials from that model can be selected. If the argument is a Flag then only materials that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any materials can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of materials selected or null if menu cancelled

Example

To select materials from model m, flagging those selected with flag f, giving the prompt 'Select materials':

```
Material.Select(f, 'Select materials', m);
```

To select materials, flagging those selected with flag f but limiting selection to materials flagged with flag l, giving the prompt 'Select materials':

```
Material.Select(f, 'Select materials', l);
```

SetErosionPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*, idam_index (optional)*[integer]*)

Description

Sets the value of Erosion property string *acronym* for this Material object

Arguments

Name	Type	Description
acronym	string	The acronym of the property value to set
value	integer/real for numeric properties, string for character properties	The value of the property to set.
idam_index (optional)	integer	Required if property is one of IDAM cards pair property (for IDAM value less than zero) . If the argument is not given, set the property values for first IDAM cards Pair. The index value starts from zero.

Return type

No return value

Example

To set the value of IDAM Erosion for material m to be 8:

```
m.SetErosionPropertyByName("idam", 8);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the material.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the material

Return type

No return value

Example

To set flag f for material m:

```
m.SetFlag(f);
```

SetMaterialErosion()

Description

Initializes the Erosion properties for this Material object.

Arguments

No arguments

Return type

No return value

Example

To set the Erosion Property for material m:

```
m.SetMaterialErosion();
```

SetPropertyByIndex(index*[integer]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of property at index *index* for this Material object

Arguments

Name	Type	Description
index	integer	The index of the property value to set. (the number of properties can be found from properties) **Note that indices start at 0**. There is no link between indices and rows/columns so adjacent fields on a line for a material may not have adjacent indices.
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the property at index 2, for material m to be 1.234:

```
m.SetPropertyByIndex(2, 1.234);
```

SetPropertyByName(acronym*[string]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of property string *acronym* for this Material object

Arguments

Name	Type	Description
acronym	string	The acronym of the property value to set
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the value of RO for material m to be 7.89e-9:

```
m.SetPropertyByName("RO", 7.89e-9);
```

SetPropertyByRowCol(row*[integer]*, col*[integer]*, value*[integer/real for numeric properties, string for character properties]*)

Description

Sets the value of the property for row and col for this Material object.**Note that rows and columns start at 0**.

Arguments

Name	Type	Description
row	integer	The row of the property value to set
col	integer	The column of the property value to set
value	integer/real for numeric properties, string for character properties	The value of the property to set.

Return type

No return value

Example

To set the value of the property at row 0, column 1 for material m to be 7.89e-9:

```
m.SetPropertyByRowCol(0, 1, 7.89e-9);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the material. The material will be sketched until you either call Material.Unsketch(), Material.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the material is sketched. If omitted redraw is true. If you want to sketch several materials and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch material m:

```
m.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged materials in the model. The materials will be sketched until you either call Material.Unsketch(), Material.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged materials will be sketched in
flag	Flag	Flag set on the materials that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the materials are sketched. If omitted redraw is true. If you want to sketch flagged materials several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all materials flagged with flag in model m:

```
Material.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of materials in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing materials should be counted. If false or omitted referenced but undefined materials will also be included in the total.

Return type

number of materials

Example

To get the total number of materials in model m:

```
var total = Material.Total(m);
```

Unblank()

Description

Unblanks the material

Arguments

No arguments

Return type

No return value

Example

To unblank material m:

```
m.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that all materials will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the materials in model m:

```
Material.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged materials will be unblanked in
flag	Flag	Flag set on the materials that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the materials in model m flagged with f:

```
Material.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all materials will be unset in
flag	Flag	Flag to unset on the materials

Return type

No return value

Example

To unset the flag f on all the materials in model m:

```
Material.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the material.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the material is unsketched. If omitted redraw is true. If you want to unsketch several materials and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch material m:

```
m.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all materials.

Arguments

Name	Type	Description
Model	Model	Model that all materials will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the materials are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all materials in model m:

```
Material.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged materials in the model.

Arguments

Name	Type	Description
Model	Model	Model that all materials will be unsketched in
flag	Flag	Flag set on the materials that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the materials are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all materials flagged with flag in model m:

```
Material.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Material object.

Example

To check if Material property m.example is a parameter by using the Material.GetParameter() method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for material. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for material m:

```
m.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this material.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for material m:

```
var xrefs = m.Xrefs();
```

YieldStress()

Description

Get Yield stress for the material.

Arguments

No arguments

Return type

real

Example

To get Yield stress for material m:

```
var yield = m.YieldStress();
```

YoungsModulus()

Description

Get Youngs modulus for the material.

Arguments

No arguments

Return type

real

Example

To get Youngs modulus for material m:

```
var e = m.YoungsModulus();
```

---

toString()

Description

Creates a string containing the material data in keyword format. Note that this contains the keyword header and the keyword cards. See also Material.Keyword() and Material.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for material m in keyword format

```
var s = m.toString();
```

Node class

The Node class gives you access to node cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Merge(Model*[Model]*, flag*[Flag]*, dist*[real]*, label (optional)*[integer]*, position (optional)*[integer]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetAttachedShells(recursive (optional)*[boolean]*)
- GetComments()
- GetFreeEdgeNodes()
- GetInitialVelocities()
- GetParameter(prop*[string]*)
- GetReferenceGeometry()
- Keyword()
- KeywordCards()
- Next()
- NodalMass()
- Previous()
- SetFlag(flag*[Flag]*)

- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Node constants

Name	Description
Node.SCALAR	Node is *NODE_SCALAR.
Node.SCALAR_VALUE	Node is *NODE_SCALAR_VALUE.

Node properties

Name	Type	Description
colour	Colour	The colour of the node
exists	logical	true if node exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the node is in.
label	integer	Node number. Also see the nid property which is an alternative name for this.
model	integer	The Model number that the node is in.
ndof	integer	Number of degrees of freedom (SCALAR and SCALAR_VALUE only).
nid	integer	Node number. Also see the label property which is an alternative name for this.
rc	integer	Rotational constraint (0-7)
scalar	integer	The type of the node. Can be false (*NODE), Node.SCALAR (*NODE_SCALAR) or Node.SCALAR_VALUE (*NODE_SCALAR_VALUE)
tc	integer	Translational constraint (0-7)
x	real	X coordinate
x1	integer	Initial value of 1st degree of freedom (SCALAR_VALUE only).
x2	integer	Initial value of 2nd degree of freedom (SCALAR_VALUE only).
x3	integer	Initial value of 3rd degree of freedom (SCALAR_VALUE only).
y	real	Y coordinate
z	real	Z coordinate

Detailed Description

The Node class allows you to create, modify, edit and manipulate node cards. See the documentation below for more details.

Constructor

new Node(Model*[Model]*, nid*[integer]*, x*[real]*, y*[real]*, z*[real]*, tc (optional)*[integer]*, rc (optional)*[integer]*)

Description

Create a new Node object.

Arguments

Name	Type	Description
Model	Model	Model that node will be created in
nid	integer	Node number
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate
tc (optional)	integer	Translational constraint (0-7). If omitted tc will be set to 0.
rc (optional)	integer	Rotational constraint (0-7). If omitted rc will be set to 0.

Return type

Node object

Example

To create a new node in model m with label 100, at coordinates (20, 40, 10)

```
var n = new Node(m, 100, 20, 40, 10);
```

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the node

Return type

No return value

Example

To associate comment c to the node n:

```
n.AssociateComment(c);
```

Blank()

Description

Blanks the node

Arguments

No arguments

Return type

No return value

Example

To blank node n:

```
n.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodes will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodes in model m:

```
Node.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodes will be blanked in
flag	Flag	Flag set on the nodes that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the nodes in model m flagged with f:

```
Node.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the node is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if node n is blanked:

```
if (n.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse node n:

```
n.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the node.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the node

Return type

No return value

Example

To clear flag f for node n:

```
n.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the node.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Node object

Example

To copy node n into node z:

```
var z = n.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a node.

Arguments

Name	Type	Description
Model	Model	Model that the node will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Node object (or null if not made)

Example

To start creating a node in model m:

```
var n = Node.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a node.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the node

Return type

No return value

Example

To detach comment c from the node n:

```
n.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit node n:

```
n.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for node n:

```
n.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for node.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the node colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the node.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing node n:

```
var colour = n.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first node in

Return type

Node object (or null if there are no nodes in the model).

Example

To get the first node in model m:

```
var n = Node.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free node label in the model. Also see Node.LastFreeLabel(), Node.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free node label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Node label.

Example

To get the first free node label in model m:

```
var label = Node.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the nodes in the model with a defined flag.

Arguments

Name	Type	Description

Model	Model	Model that all nodes will be flagged in
flag	Flag	Flag to set on the nodes

Return type

No return value

Example

To flag all of the nodes with flag f in model m:

```
Node.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the node is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the node

Return type

true if flagged, false if not.

Example

To check if node n has flag f set on it:

```
if (n.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each node in the model.
Note that ForEach has been designed to make looping over nodes as fast as possible and so has some limitations. Firstly, a single temporary Node object is created and on each function call it is updated with the current node data. This means that you should not try to store the Node object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new nodes inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all nodes are in
func	function	Function to call for each node
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the nodes in model m:

```
Node.ForEach(m, test);
function test(n)
{
// n is Node object
}
```

To call function test for all of the nodes in model m with optional object:

```
var data = { x:0, y:0 };
Node.ForEach(m, test, data);
function test(n, extra)
{
// n is Node object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Node objects for all of the nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodes from

Return type

Array of Node objects

Example

To make an array of Node objects for all of the nodes in model m

```
var n = Node.GetAll(m);
```

GetAttachedShells(recursive (optional)*[boolean]*)

Description

Returns the shells that are attached to the node.

Arguments

Name	Type	Description
recursive (optional)	boolean	If recursive is false then only the shells actually attached to the node will be returned (this could also be done by using the Xrefs class but this method is provided for convenience. If recursive is true then PRIMER will keep finding attached shells until no more can be found. If omitted recursive will be false.

Return type

Array of Shell objects (or null if there are no attached shells).

Example

To find the shells attached to node n, growing the selection until no more shells can be found:

```
var shell_array = n.GetAttachedShells(true);
```

GetComments()

Description

Extracts the comments associated to a node.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the node n:

```
var comm_array = n.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Node objects for all of the flagged nodes in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get nodes from
flag	Flag	Flag set on the nodes that you want to retrieve

Return type

Array of Node objects

Example

To make an array of Node objects for all of the nodes in model m flagged with f

```
var n = Node.GetFlagged(m, f);
```

GetFreeEdgeNodes()

Description

If the node is on a shell free edge and that edge forms a loop like the boundary of a hole, then GetFreeEdgeNodes returns all of the nodes on the hole/boundary in order.

Note that a free edge is a shell edge which is only used by one shell, whereas edges in the middle of a shell part will have got more than one adjacent shell and are therefore not free edges. If every node on a boundary belongs to exactly two free edges, then this function returns the array as described. In more involved combinatorics of shells, for example multiple parts sharing nodes along their boundaries, there can be one, three or more free edges at a node, and this function should not be used.

If you only need to know whether or not a node is on a free edge, you should find the shells attached to it by cross references with Xrefs.GetItemID and see whether these shells have got other nodes in common as well. If nodes along an edge of a shell only appear in that one shell, this is a free edge.

Arguments

No arguments

Return type

Array of Node objects (or null if not on a shell free edge).

Example

To find all the nodes on the hole/boundary that node n is on:

```
var node_array = n.GetFreeEdgeNodes();
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Node object for a node ID.

Arguments

Name	Type	Description
Model	Model	Model to find the node in
number	integer	number of the node you want the Node object for

Return type

Node object (or null if node does not exist).

Example

To get the Node object for node 100 in model m

```
var n = Node.GetFromID(m, 100);
```

GetInitialVelocities()

Description

Returns the initial velocity of the node. You need to be sure the field nvels of the node is populate before to use GetInitialVelocities. To do so you can use Model.PopNodeVels .

Arguments

No arguments

Return type

Array containing the 3 translational and 3 rotational velocity values.

Example

To get the initial velocity of the node n:

```
var vel = n.GetInitialVelocities();
```

GetParameter(prop*[string]*)

Description

Checks if a Node property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Node.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	node property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Node property n.example is a parameter:

```
Options.property_parameter_names = true;
if (n.GetParameter(n.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Node property n.example is a parameter by using the GetParameter method:

```
if (n.ViewParameters().GetParameter(n.example) ) do_something...
```

GetReferenceGeometry()

Description

Returns the airbag reference geometry of the node

Arguments

No arguments

Return type

The reference geometry ID of the node (or 0 if it hasn't got any)

Example

To get the reference geometry of the node n:

```
var a = n.GetReferenceGeometry();
```

Keyword()

Description

Returns the keyword for this node (*NODE, *NODE_SCALAR or *NODE_SCALAR_VALUE). **Note that a carriage return is not added**. See also Node.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for node n:

```
var key = n.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the node. **Note that a carriage return is not added**. See also Node.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for node n:

```
var cards = n.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last node in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last node in

Return type

Node object (or null if there are no nodes in the model).

Example

To get the last node in model m:

```
var n = Node.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free node label in the model. Also see Node.FirstFreeLabel(), Node.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free node label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Node label.

Example

To get the last free node label in model m:

```
var label = Node.LastFreeLabel(m);
```

Merge(Model*[Model]*, flag*[Flag]*, dist*[real]*, label (optional)*[integer]*, position (optional)*[integer]*) [static]

Description

Attempts to merge nodes flagged with flag for a model in PRIMER. Merging nodes on *AIRBAG_SHELL_REFERENCE_GEOMETRY can be controlled by using Options.node_replace_asrg. Also see Model.MergeNodes().

Arguments

Name	Type	Description
Model	Model	Model that the nodes will be merged in
flag	Flag	Flag set on nodes to nodes
dist	real	Nodes closer than dist will be potentially merged.
label (optional)	integer	Label to keep after merge. If > 0 then highest label kept. If <= 0 then lowest kept. If omitted the lowest label will be kept.
position (optional)	integer	Position to merge at. If > 0 then merged at highest label position. If < 0 then merged at lowest label position. If 0 then merged at midpoint. If omitted the merge will be done at the lowest label.

Return type

The number of nodes merged

Example

To (try to) merge nodes in model m flagged with flag f, with a distance of 0.1:

```
Node.Merge(m, f, 0.1);
```

Next()

Description

Returns the next node in the model.

Arguments

No arguments

Return type

Node object (or null if there are no more nodes in the model).

Example

To get the node in model m after node n:

```
var n = n.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) node label in the model. Also see Node.FirstFreeLabel(), Node.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free node label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Node label.

Example

To get the next free node label in model m:

```
var label = Node.NextFreeLabel(m);
```

NodalMass()

Description

Get the mass of a node. This will be the sum of the structural element mass attached to the node plus any lumped mass. If called on the node of a PART_INERTIA or NRBC_INERTIA, this function will return the mass of the part/nrbc, as 'nodal mass' has no meaning in this context.

Arguments

No arguments

Return type

real

Example

To get the mass for node n:

```
var mass = n.NodalMass();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a node.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodes from that model can be picked. If the argument is a Flag then only nodes that are flagged with *limit* can be selected. If omitted, or null, any nodes from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Node object (or null if not picked)

Example

To pick a node from model m giving the prompt 'Pick node from screen':

```
var n = Node.Pick('Pick node from screen', m);
```

Previous()

Description

Returns the previous node in the model.

Arguments

No arguments

Return type

Node object (or null if there are no more nodes in the model).

Example

To get the node in model m before node n:

```
var n = n.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodes will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the nodes in model m, from 1000000:

```
Node.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodes will be renumbered in
flag	Flag	Flag set on the nodes that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the nodes in model m flagged with f, from 1000000:

```
Node.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select nodes using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting nodes
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only nodes from that model can be selected. If the argument is a Flag then only nodes that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any nodes can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of nodes selected or null if menu cancelled

Example

To select nodes from model m, flagging those selected with flag f, giving the prompt 'Select nodes':

```
Node.Select(f, 'Select nodes', m);
```

To select nodes, flagging those selected with flag f but limiting selection to nodes flagged with flag l, giving the prompt 'Select nodes':

```
Node.Select(f, 'Select nodes', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the node.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the node

Return type

No return value

Example

To set flag f for node n:

```
n.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the node. The node will be sketched until you either call Node.Unsketch(), Node.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the node is sketched. If omitted redraw is true. If you want to sketch several nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch node n:

```
n.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged nodes in the model. The nodes will be sketched until you either call Node.Unsketch(), Node.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged nodes will be sketched in
flag	Flag	Flag set on the nodes that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the nodes are sketched. If omitted redraw is true. If you want to sketch flagged nodes several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all nodes flagged with flag in model m:

```
Node.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing nodes should be counted. If false or omitted referenced but undefined nodes will also be included in the total.

Return type

number of nodes

Example

To get the total number of nodes in model m:

```
var total = Node.Total(m);
```

Unblank()

Description

Unblanks the node

Arguments

No arguments

Return type

No return value

Example

To unblank node n:

```
n.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the nodes in the model.

Arguments

Name	Type	Description

Model	Model	Model that all nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodes in model m:

```
Node.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged nodes will be unblanked in
flag	Flag	Flag set on the nodes that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the nodes in model m flagged with f:

```
Node.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all nodes will be unset in
flag	Flag	Flag to unset on the nodes

Return type

No return value

Example

To unset the flag f on all the nodes in model m:

```
Node.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the node.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the node is unsketched. If omitted redraw is true. If you want to unsketch several nodes and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch node n:

```
n.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all nodes.

Arguments

Name	Type	Description
Model	Model	Model that all nodes will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodes in model m:

```
Node.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged nodes in the model.

Arguments

Name	Type	Description
Model	Model	Model that all nodes will be unsketched in
flag	Flag	Flag set on the nodes that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the nodes are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all nodes flagged with flag in model m:

```
Node.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Node object.

Example

To check if Node property n.example is a parameter by using the Node.GetParameter() method:

```
if (n.ViewParameters().GetParameter(n.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for node. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for node n:

```
n.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this node.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for node n:

```
var xrefs = n.Xrefs();
```

---

toString()

Description

Creates a string containing the node data in keyword format. Note that this contains the keyword header and the keyword cards. See also Node.Keyword() and Node.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for node n in keyword format

```
var s = n.toString();
```

Parameter class

The Parameter class allows you to access the parameters in a model. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- AutoReorder(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- GetAll(Model*[Model]*)
- GetAllOfName(Model*[Model]*)
- GetFromName(Model*[Model]*, parameter name*[string]*)
- SaveAll(Model*[Model]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UpdateAll(Model*[Model]*)

Member functions

- ClearFlag(flag*[Flag]*)
- Error(message*[string]*, details (optional)*[string]*)
- Evaluate()
- Flagged(flag*[Flag]*)
- Keyword()
- KeywordCards()
- SetFlag(flag*[Flag]*)
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Parameter constants

Name	Description
Parameter.CHARACTER	Parameter is a character.
Parameter.INTEGER	Parameter is an integer.
Parameter.LOCAL	Parameter has _LOCAL suffix (used in suffix argument for constructor).
Parameter.MUTABLE	Parameter has _MUTABLE suffix (used in suffix argument for constructor).
Parameter.REAL	Parameter is a real.

Parameter properties

Name	Type	Description
expression (read only)	logical	true if this parameter is a *PARAMETER_EXPRESSION, false otherwise.
include	integer	The Include file number that the parameter is in.
local	logical	true if this parameter is a *PARAMETER_... _LOCAL, false otherwise.
model	integer	The Model number that the parameter is in.

mutable	logical	true if this parameter is a *PARAMETER_... _MUTABLE, false otherwise.
name (read only)	string	Parameter name.
type (read only)	constant	Can be Parameter.INTEGER, Parameter.REAL or Parameter.CHARACTER.
value	integer/real/string	Parameter value. The value will be a string for parameter expressions, or a number for normal parameters. By default when a parameter value is changed PRIMER will re-evaluate and update all of the parameters in the model as changing this parameter could cause others to change because of parameter expressions. There could be some situations where changing parameters one at a time could cause problems with re-evaluation. For example, changing parameter A could temporarily cause a division by zero when re-evaluating parameter expression B until parameter C is changed. In this case the automatic re-evaluation can be prevented by using Parameter.SaveAll and Parameter.UpdateAll.

Detailed Description

The Parameter class allows to create and query parameters in a model. See the documentation below for more details.

Constructor

new Parameter(Model*[Model]*, name*[string]*, type*[constant]*, expression*[boolean]*, value*[integer/real/string]*, suffix (optional)*[constant]*)

Description

Create a new Parameter object.

Arguments

Name	Type	Description
Model	Model	Model that parameter will be created in
name	string	Parameter name
type	constant	Can be Parameter.INTEGER, Parameter.REAL or Parameter.CHARACTER.
expression	boolean	true if *PARAMETER_EXPRESSION, false otherwise.
value	integer/real/string	Parameter value. The value will be a string for character parameters or parameter expressions, or a number for integer or real parameters.
suffix (optional)	constant	Keyword suffix Can be Parameter.LOCAL for *PARAMETER_..._LOCAL, Parameter.MUTABLE for *PARAMETER_..._MUTABLE. These may be bitwise ORed together, ie Parameter.LOCAL | Parameter.MUTABLE. If omitted the parameter will not be local or mutable.

Return type

Parameter object

Example

To create a new real parameter THK in model m with value 5.0

```
var p = new Parameter(m, "THK", Parameter.REAL, false, 5.0);
```

To create a new LOCAL integer parameter INDEX in model m with value 3

```
var p = new Parameter(m, "INDEX", Parameter.INTEGER, false, 3, Parameter.LOCAL);
```

Details of functions

AutoReorder(Model*[Model]*) [static]

Description

Auto Reorders all the parameters in the model.

Arguments

Name	Type	Description
Model	Model	Model that contains all parameters that will be re-ordered

Return type

No return value

Example

To auto-reorder all parameters in model m:

```
Parameter.AutoReorder(m);
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the parameter.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the parameter

Return type

No return value

Example

To clear flag f for parameter p:

```
p.ClearFlag(f);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for a parameter. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for parameter p:

```
p.Error("My custom error");
```

Evaluate()

Description

Evaluates a parameter expression, updating the evaluated value stored in PRIMER and returns the value. If the parameter is not an expression then the parameter value will just be returned.
If evaluating the expression cannot be done because of an error (e.g. dividing by zero) an exception will be thrown.

Arguments

No arguments

Return type

number (real and integer parameters) or string (character parameters)

Example

To evaluate parameter p:

```
var value = p.Evaluate();
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the parameters in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all parameters will be flagged in
flag	Flag	Flag to set on the parameters

Return type

No return value

Example

To flag all of the parameters with flag f in model m:

```
Parameter.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the parameter is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the parameter

Return type

true if flagged, false if not.

Example

To check if parameter p has flag f set on it:

```
if (p.Flagged(f) ) do_something...
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Parameter objects for all of the parameters in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get parameters from

Return type

Array of Parameter objects

Example

To make an array of Parameter objects for all of the parameters in model m

```
var p = Parameter.GetAll(m);
```

GetAllOfName(Model*[Model]*) [static]

Description

Returns an array of Parameter objects for all parameters in a model matching Name. If none are found that match it will return NULL. (Multiple parameters of the same name may exist if they use the _LOCAL or _MUTABLE suffices. PRIMER will also store multiple illegal instances of parameter name, using the instance as determined by the PARAMETER_DUPLICATION card.)

Arguments

Name	Type	Description
Model	Model	Model to get parameters from

Return type

Array of Parameter objects

Example

To make an array of Parameter objects for all of the parameters of name in model m

```
var p = Parameter.GetAllOfName(m, name);
```

GetFromName(Model*[Model]*, parameter name*[string]*) [static]

Description

Returns the stored Parameter object for a parameter name. WARNING: if more than one parameter Name exists (eg _ LOCAL, _MUTABLE) then only the first occurrence is returned. To return all parameters matching Name use GetAllOfName() instead.

Arguments

Name	Type	Description
Model	Model	Model to find the parameter in
parameter name	string	name of the parameter you want the Parameter object for

Return type

Parameter object (or null if parameter does not exist).

Example

To get the Parameter object for parameter "THK" in model m

```
var p = Parameter.GetFromName(m, "THK");
```

Keyword()

Description

Returns the keyword for this parameter (*PARAMETER, *PARAMETER_EXPRESSION). **Note that a carriage return is not added**. See also Parameter.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for parameter p:

```
var key = p.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the parameter. **Note that a carriage return is not added**. See also Parameter.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for parameter p:

```
var cards = p.KeywordCards();
```

SaveAll(Model[*Model*]) [static]

Description

Saves the current status and values of all of the parameters in the model. Calling this will also have the effect of turning off re-evaluating and updating of all parameters in the model when a parameter value is changed.
To update several parameters in a model without re-evaluating all the parameters after each one is changed first call this, then update all of the parameter values, and then call Parameter.UpdateAll to apply the update.
Parameter.SaveAll **must** be called before using Parameter.UpdateAll.

Arguments

Name	Type	Description
Model	Model	Model that the parameters will be saved in

Return type

No return value

Example

To save the status of all of the parameters in model m:

```
Parameter.SaveAll(m);
```

SetFlag(flag[*Flag*])

Description

Sets a flag on the parameter.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the parameter

Return type

No return value

Example

To set flag f for parameter p:

```
p.SetFlag(f);
```

UnflagAll(Model[*Model*], flag[*Flag*]) [static]

Description

Unsets a defined flag on all of the parameters in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all parameters will be unset in
flag	Flag	Flag to unset on the parameters

Return type

No return value

Example

To unset the flag f on all of the parameters in model m:

```
Parameter.UnflagAll(m, f);
```

UpdateAll(Model*[Model]*) [static]

Description

Updates all of the parameters in the model after saving the state of all parameters using Parameter.SaveAll and modifying the parameter values. As parameter re-evaluation has been suppressed by Parameter.SaveAll you should ensure that all parameters in the model can be evaluated correctly before calling this to ensure that there are no errors. If any of the parameters cannot be evaluated then the values saved in Parameter.SaveAll will be restored, the update will be aborted and an exception thrown. Calling this will also have the effect of turning back on re-evaluating and updating of all parameters in the model when a parameter value is changed.
Parameter.SaveAll **must** be called before this method can be used.

Arguments

Name	Type	Description
Model	Model	Model that the parameters will be updated in

Return type

No return value

Example

To update all of the parameters in model m:

```
Parameter.UpdateAll(m);
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for a parameter. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for parameter p:

```
p.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this parameter.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for parameter p:

```
var xrefs = p.Xrefs();
```

---

toString()

Description

Creates a string containing the parameter data in keyword format. Note that this contains the keyword header and the keyword cards. See also Parameter.Keyword() and Parameter.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for parameter p in keyword format

```
var s = p.toString();
```

Part class

The Part class gives you access to part cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- AllTableProperties(Model*[Model]*)
- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- FlagVisible(Model*[Model]*, flag*[Flag]*)
- FlaggedTableProperties(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- MeasurePartToPart(part1*[Part]*, part2*[Part]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- CentreOfGravity(options (optional)*[object]*)
- CentreOfGravity(option (optional)*[boolean]*) [deprecated]
- ClearFlag(flag*[Flag]*)
- ClosestNode(x*[real]*, y*[real]*, z*[real]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetCompositeData(ipt*[integer]*)
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Mass()

- MaxMin()
- Next()
- Previous()
- RemoveCompositeData(ipt*[integer]*)
- SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*, tmid (optinal)*[integer]*, plyid (optional)*[integer]*, shrfac (optional)*[real]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- TableProperties()
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Part properties

Name	Type	Description
adpopt	integer	Adaptivity flag
ansid	integer	Attachment node set ID
attachment_ nodes	logical	If _ATTACHMENT_NODES option is set. Can be true or false
averaged	logical	If _AVERAGED option is set. Can be true or false
cadname	string	CAD name stored for Part (or null if doesn't exist). This property is only used by PRIMER.
cid	integer	Coordinate system number
cmsn	integer	CAL3D/MADYMO number
colour	Colour	The colour of the part
composite	logical	If _COMPOSITE option is set. Can be true or false
composite_long	logical	If _COMPOSITE_LONG option is set. Can be true or false
contact	logical	If _CONTACT option is set. Can be true or false
dc	real	Exponential decay coefficient
element_type	string	The type of elements the Part contains. e.g. "SHELL", "SOLID" or null if empty/no section (read only).
elform	integer	Element formulation
eosid	integer or string	Equation of state number or character label
exists	logical	true if part exists, false if referred to but not defined. (read only)
fd	real	Dynamic coefficient of friction
fs	real	Static coefficient of friction
grav	integer	Gravity loading
heading	string	Part heading
hgid	integer or string	Hourglass number or character label
hmname	string	Hypermesh comment read from keyword file for Part (or null if doesn't exist).
include	integer	The Include file number that the part is in.
inertia	logical	If _INERTIA option is set. Can be true or false

ircs	integer	Flag for inertia tensor reference coordinate system
ixx	real	Ixx component of inertia tensor
ixy	real	Ixy component of inertia tensor
ixz	real	Ixz component of inertia tensor
iyy	real	Iyy component of inertia tensor
iyz	real	Iyz component of inertia tensor
izz	real	Izz component of inertia tensor
label	integer or string	Part number or character label. Also see the pid property which is an alternative name for this.
marea	real	Non structural mass per unit area
mdep	integer	MADYMO ellipse/plane number
mid	integer or string	Material number or character label
model	integer	The Model number that the part is in.
movopt	integer	Flag to deactivate moving for merged rigid bodies
nip	integer	Number of integration points (layers) present for _COMPOSITE parts
nloc	integer	Location of reference surface
nodeid	integer	Node ID for centre of rigid body
optt	real	Contact thickness
pid	integer or string	Part number or character label. Also see the label property which is an alternative name for this.
prbf	integer	Print flag for RBDOUT and MATSUM files
print	logical	If _PRINT option is set. Can be true or false
reposition	logical	If _REPOSITION option is set. Can be true or false
rigid	logical	true if part is rigid, false if deformable. (read only)
secid	integer or string	Section number or character label
sft	real	Thickness scale factor
shrf	real	Shear correction factor
ssf	real	Scale factor on default slave penalty stiffness
thshel	integer	Thermal shell formulation
tm	real	total mass
tmid	integer or string	Thermal material number or character label
transparency	integer	The transparency of the part (0-100) 0% is opaque, 100% is transparent.
tshear	integer	Flag for transverse shear strain distribution
tshell	logical	If _COMPOSITE_TSHELL option is set. Can be true or false
vc	real	Coefficient for viscous friction
vrx	real	x rotational velocity
vry	real	y rotational velocity
vrz	real	z rotational velocity

vtx	real	x translational velocity
vty	real	y translational velocity
vtz	real	z translational velocity
xc	real	x coordinate of centre of mass
xl	real	x coordinate of local x axis
xlip	real	x coordinate of vector in local xy plane
yc	real	y coordinate of centre of mass
yl	real	y coordinate of local x axis
ylip	real	y coordinate of vector in local xy plane
zc	real	z coordinate of centre of mass
zl	real	z coordinate of local x axis
zlip	real	z coordinate of vector in local xy plane

Detailed Description

The Part class allows you to create, modify, edit and manipulate part cards. See the documentation below for more details.

Constructor

new Part(Model*[Model]*, pid*[integer or string]*, secid*[integer or string]*, mid*[integer or string]*, heading (optional)*[string]*)

Description

Create a new Part object.

Arguments

Name	Type	Description
Model	Model	Model that part will be created in
pid	integer or string	Part number or character label
secid	integer or string	Section number or character label
mid	integer or string	Material number or character label
heading (optional)	string	Title for the part

Return type

Part object

Example

To create a new part called 'Example' in model m with label 100, section 1, material 10:

```
var p = new Part(m, 100, 1, 10, 'Example');
```

Details of functions

AllTableProperties(Model[*Model*]) [static]

Description

Returns all of the properties available in the part table for the parts. The table values are returned in an array of objects (an object for each part). The object property names are the same as the table headers but spaces are replaced with underscore characters and characters other than 0-9, a-z and A-Z are removed to ensure that the property name is valid in JavaScript. If a table value is undefined the property value will be the JavaScript undefined value. If the table value is a valid number it will be a number, otherwise the value will returned as a string.

Arguments

Name	Type	Description
Model	Model	Model that the flagged parts are in

Return type

Array of objects

Example

To get all of the properties for parts in model m:

```
var properties = Part.AllTableProperties(m);
for (var p=0; p<properties.length; p++)
{
    for (var x in properties[p])
    {
        Message(x+"="+properties[p][x]);
    }
}
```

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the part

Return type

No return value

Example

To associate comment c to the part p:

```
p.AssociateComment(c);
```

Blank()

Description

Blanks the part

Arguments

No arguments

Return type

No return value

Example

To blank part p:

```
p.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the parts in model m:

```
Part.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged parts will be blanked in
flag	Flag	Flag set on the parts that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the parts in model m flagged with f:

```
Part.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the part is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if part p is blanked:

```
if (p.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse part p:

```
p.Browse();
```

CentreOfGravity(options (optional)*[object]*)

Description

Returns the centre of gravity for a part.

Arguments

Name	Type	Description

<table>
<tr><td>options (optional)</td><td>object</td><td>Options specifying how the mass calculation should be done. Object has the following
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>lumpedmass (optional)</td><td>boolean</td><td>Lumped mass is included for deformable parts. Off by default.</td></tr>
<tr><td>nrbmass (optional)</td><td>boolean</td><td>NRB mass is included for deformable parts. Off by default. (tranfermass:true required for this option)</td></tr>
<tr><td>plot (optional)</td><td>boolean</td><td>Plot CofG.</td></tr>
<tr><td>skipslave (optional)</td><td>boolean</td><td>Slave rigid part is assigned zero mass (if slaveparts = true). On by default.</td></tr>
<tr><td>slaveparts (optional)</td><td>boolean</td><td>Mass of rigid master part includes mass of its slave parts. On by default.</td></tr>
<tr><td>timestepmass (optional)</td><td>boolean</td><td>Timestep added mass is included for deformable parts. Off by default.</td></tr>
<tr><td>transfermass (optional)</td><td>boolean</td><td>Mass of deformable nodes attached to rigid part/nrb is transferred. On by default.</td></tr>
</table>
properties:</td></tr>
</table>

Return type

An array containing the x, y and z coordinates for the CofG.

Example

To get the centre of gravity for part p with options configured:

```
var cofg = p.CentreOfGravity({slaveparts:false, transfermass:true,
lumpedmass:false, nrbmass:true, timestepmass:false, plot:true});
var x = cofg[0];
var y = cofg[1];
var z = cofg[2];
```

CentreOfGravity(option (optional)*[boolean]*) [deprecated]

This function is deprecated in version 16.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Returns the centre of gravity for a part. Rigid parts will always include mass of slave parts.
Mass is transferred from deformable to rigid when nodes attach.

Arguments

Name	Type	Description
option (optional)	boolean	If set, centre of gravity calculation for deformable parts includes lumped mass, mass of nodal rigid bodies and timestep added mass.

Return type

An array containing the x, y and z coordinates for the CofG.

Example

To get the centre of gravity for part p:

```
var cofg = p.CentreOfGravity();
           var x = cofg[0];
           var y = cofg[1];
           var z = cofg[2];
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the part.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the part

Return type

No return value

Example

To clear flag f for part p:

```
p.ClearFlag(f);
```

ClosestNode(x*[real]*, y*[real]*, z*[real]*)

Description

Finds the Node on the part closest to a coordinate.

Arguments

Name	Type	Description
x	real	X coordinate of point
y	real	Y coordinate of point
z	real	Z coordinate of point

Return type

ID of Node or null if part has no nodes

Example

To find the node on part p closest to point (1, 2, 3):

```
var n = p.ClosestNode(1, 2, 3);
```

Copy(range (optional)*[boolean]*)

Description

Copies the part.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Part object

Example

To copy part p into part z:

```
var z = p.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a part.

Arguments

Name	Type	Description
Model	Model	Model that the part will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Part object (or null if not made)

Example

To start creating a part in model m:

```
var p = Part.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a part.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the part

Return type

No return value

Example

To detach comment c from the part p:

```
p.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit part p:

```
p.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for part p:

```
p.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for part.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the part colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the part.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing part p:

```
var colour = p.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first part in

Return type

Part object (or null if there are no parts in the model).

Example

To get the first part in model m:

```
var p = Part.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free part label in the model. Also see Part.LastFreeLabel(), Part.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free part label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Part label.

Example

To get the first free part label in model m:

```
var label = Part.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the parts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be flagged in
flag	Flag	Flag to set on the parts

Return type

No return value

Example

To flag all of the parts with flag f in model m:

```
Part.FlagAll(m, f);
```

FlagVisible(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all the unblanked parts in the model.

Arguments

Name	Type	Description
Model	Model	Model for which all unblanked parts will be flagged in
flag	Flag	Flag to set on the unblanked parts

Return type

No return value

Example

To flag all unblanked parts in model m with flag f:

```
Part.FlagVisible(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the part is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the part

Return type

true if flagged, false if not.

Example

To check if part p has flag f set on it:

```
if (p.Flagged(f) ) do_something...
```

FlaggedTableProperties(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns all of the properties available in the part table for the flagged parts. The table values are returned in an array of objects (an object for each part). The object property names are the same as the table headers but spaces are replaced with underscore characters and characters other than 0-9, a-z and A-Z are removed to ensure that the property name is valid in JavaScript. If a table value is undefined the property value will be the JavaScript undefined value. If the table value is a valid number it will be a number, otherwise the value will returned as a string.

Arguments

Name	Type	Description
Model	Model	Model that the flagged parts are in
flag	Flag	Flag set on the parts that you want properties for

Return type

Array of objects

Example

To get all of the properties for parts in model m flagged with f:

```
var properties = Part.FlaggedTableProperties(m, f);
for (var p=0; p<properties.length; p++)
{
    for (var x in properties[p])
    {
        Message(x+"="+properties[p][x]);
    }
}
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each part in the model.
Note that ForEach has been designed to make looping over parts as fast as possible and so has some limitations. Firstly, a single temporary Part object is created and on each function call it is updated with the current part data. This means that you should not try to store the Part object for later use (e.g. in an array) as it is temporary. Secondly, you cannot create new parts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all parts are in
func	function	Function to call for each part
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the parts in model m:

```
Part.ForEach(m, test);
function test(p)
{
// p is Part object
}
```

To call function test for all of the parts in model m with optional object:

```
var data = { x:0, y:0 };
Part.ForEach(m, test, data);
function test(p, extra)
{
// p is Part object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Part objects for all of the parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get parts from

Return type

Array of Part objects

Example

To make an array of Part objects for all of the parts in model m

```
var p = Part.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a part.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the part p:

```
var comm_array = p.GetComments();
```

GetCompositeData(ipt*[integer]*)

Description

Returns the composite data for an integration point in *PART_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want the data for. **Note that integration points start at 0, not 1**.

Return type

An array containing the material id, thickness, beta angle and thermal material values. If the _COMPOSITE_LONG option is set, then the array returned will also contain the ply ID.

Example

To get the composite data for the 3rd integration point for part p:

```
if (p.composite && p.nip >= 3)
{
    var ipt_data = p.GetCompositeData(2);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Part objects for all of the flagged parts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get parts from
flag	Flag	Flag set on the parts that you want to retrieve

Return type

Array of Part objects

Example

To make an array of Part objects for all of the parts in model m flagged with f

```
var p = Part.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Part object for a part ID.

Arguments

Name	Type	Description
Model	Model	Model to find the part in
number	integer	number of the part you want the Part object for

Return type

Part object (or null if part does not exist).

Example

To get the Part object for part 100 in model m

```
var p = Part.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Part property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Part.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	part property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Part property p.example is a parameter:

```
Options.property_parameter_names = true;
if (p.GetParameter(p.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Part property p.example is a parameter by using the GetParameter method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this part (*PART, *PART_SCALAR or *PART_SCALAR_VALUE). **Note that a carriage return is not added**. See also Part.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for part p:

```
var key = p.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the part. **Note that a carriage return is not added**. See also Part.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for part p:

```
var cards = p.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last part in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last part in

Return type

Part object (or null if there are no parts in the model).

Example

To get the last part in model m:

```
var p = Part.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free part label in the model. Also see Part.FirstFreeLabel(), Part.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free part label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Part label.

Example

To get the last free part label in model m:

```
var label = Part.LastFreeLabel(m);
```

Mass()

Description

Returns the mass properties for a part.

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
Assign_Mass	real	Assign mass
Component_Mass	real	Component mass
Dyna_Added_Mass	real	Dyna added mass
Dyna_Part_Mass	real	Dyna part mass
Lumped_Mass	real	Lumped mass
NRB_Mass	real	NRB mass
NS_Mass	real	Non-structural mass
Struct_Mass	real	Structural mass
Transferrd_Mass	real	Transferred mass when deformable meshed to rigid

Example

To get the structural mass for part p:

```
var mprops = p.Mass();
var struct_mass = mprops.Struct_Mass;
```

MaxMin()

Description

Returns the max and min boundas of a part

Arguments

No arguments

Return type

An array containing the xMin, xMax, yMin, yMax, zMin and zMax coordinates for a box bounding the part.

Example

To get the bounds for part p:

```
var bounds = p.MaxMin();
var xMin = bounds[0];
var xMax = bounds[1];
var yMin = bounds[2];
var yMax = bounds[3];
var zMin = bounds[4];
var zMax = bounds[5];
```

MeasurePartToPart(part1[*Part*], part2[*Part*]) [static]

Description

This static method measures the distance between two part objects contained in the same model or in two different models

Arguments

Name	Type	Description
part1	Part	Part to measure from
part2	Part	Part to measure to

Return type

Object with the following properties:

Name	Type	Description
distance	real	Distance between the two parts
vector	Array of reals	Components of distance vector

Example

To measure the distance between part object p1 and part object p2:

```
var m = Part.MeasurePartToPart(p1, p2);
var d = m.distance;
var XComp = m.vector[0];
var YComp = m.vector[1];
var ZComp = m.vector[2];
```

Next()

Description

Returns the next part in the model.

Arguments

No arguments

Return type

Part object (or null if there are no more parts in the model).

Example

To get the part in model m after part p:

```
var p = p.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) part label in the model. Also see Part.FirstFreeLabel(), Part.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free part label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Part label.

Example

To get the next free part label in model m:

```
var label = Part.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a part.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only parts from that model can be picked. If the argument is a Flag then only parts that are flagged with *limit* can be selected. If omitted, or null, any parts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Part object (or null if not picked)

Example

To pick a part from model m giving the prompt 'Pick part from screen':

```
var p = Part.Pick('Pick part from screen', m);
```

Previous()

Description

Returns the previous part in the model.

Arguments

No arguments

Return type

Part object (or null if there are no more parts in the model).

Example

To get the part in model m before part p:

```
var p = p.Previous();
```

RemoveCompositeData(ipt*[integer]*)

Description

Removes the composite data for an integration point in *PART_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to remove. **Note that integration points start at 0, not 1**.

Return type

No return value.

Example

To remove the composite data for the 3rd integration point for part p:

```
p.RemoveCompositeData(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the parts in model m, from 1000000:

```
Part.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged parts will be renumbered in
flag	Flag	Flag set on the parts that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the parts in model m flagged with f, from 1000000:

```
Part.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select parts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting parts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only parts from that model can be selected. If the argument is a Flag then only parts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any parts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of parts selected or null if menu cancelled

Example

To select parts from model m, flagging those selected with flag f, giving the prompt 'Select parts':

```
Part.Select(f, 'Select parts', m);
```

To select parts, flagging those selected with flag f but limiting selection to parts flagged with flag l, giving the prompt 'Select parts':

```
Part.Select(f, 'Select parts', l);
```

SetCompositeData(ipt*[integer]*, mid*[integer]*, thick*[real]*, beta*[real]*, tmid (optinal)*[integer]*, plyid (optional)*[integer]*, shrfac (optional)*[real]*)

Description

Sets the composite data for an integration point in *PART_COMPOSITE.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to set the data for. **Note that integration points start at 0, not 1**.
mid	integer	Material ID for the integration point.
thick	real	Thickness of the integration point.
beta	real	Material angle of the integration point.
tmid (optinal)	integer	Thermal material ID for the integration point.
plyid (optional)	integer	Ply ID for the integration point. This should be used if the _COMPOSITE_LONG option is set for the part.

shrfac (optional)	real	Transverse shear stress scale factor.

Return type

No return value.

Example

To set the composite data for the 3rd integration point to mat 1, thickness 0.5 and angle 45, for part p:

```
p.SetCompositeData(2, 1, 0.5, 45);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the part.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the part

Return type

No return value

Example

To set flag f for part p:

```
p.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the part. The part will be sketched until you either call Part.Unsketch(), Part.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the part is sketched. If omitted redraw is true. If you want to sketch several parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch part p:

```
p.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged parts in the model. The parts will be sketched until you either call Part.Unsketch(), Part.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged parts will be sketched in
flag	Flag	Flag set on the parts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the parts are sketched. If omitted redraw is true. If you want to sketch flagged parts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all parts flagged with flag in model m:

```
Part.SketchFlagged(m, flag);
```

TableProperties()

Description

Returns all of the properties available for the part in the part table. The table values are returned in an object. The object property names are the same as the table headers but spaces are replaced with underscore characters and characters other than 0-9, a-z and A-Z are removed to ensure that the property name is valid in JavaScript. If a table value is undefined the property value will be the JavaScript undefined value. If the table value is a valid number it will be a number, otherwise the value will returned as a string.

Arguments

No arguments

Return type

object.

Example

To get all of the properties for part p:

```
var properties = p.TableProperties();
for (var x in properties)
{
    Message(x+"="+properties[x]);
}
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of parts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing parts should be counted. If false or omitted referenced but undefined parts will also be included in the total.

Return type

number of parts

Example

To get the total number of parts in model m:

```
var total = Part.Total(m);
```

Unblank()

Description

Unblanks the part

Arguments

No arguments

Return type

No return value

Example

To unblank part p:

```
p.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the parts in model m:

```
Part.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged parts will be unblanked in
flag	Flag	Flag set on the parts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the parts in model m flagged with f:

```
Part.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all parts will be unset in
flag	Flag	Flag to unset on the parts

Return type

No return value

Example

To unset the flag f on all the parts in model m:

```
Part.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the part.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the part is unsketched. If omitted redraw is true. If you want to unsketch several parts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch part p:

```
p.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all parts.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all parts in model m:

```
Part.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged parts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all parts will be unsketched in
flag	Flag	Flag set on the parts that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the parts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all parts flagged with flag in model m:

```
Part.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Part object.

Example

To check if Part property p.example is a parameter by using the Part.GetParameter() method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for part. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for part p:

```
p.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this part.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for part p:

```
var xrefs = p.Xrefs();
```

---

toString()

Description

Creates a string containing the part data in keyword format. Note that this contains the keyword header and the keyword cards. See also Part.Keyword() and Part.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for part p in keyword format

```
var str = p.toString();
```

Rigidwall class

The Rigidwall class gives you access to rigidwall cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- FindNodesBehind(flag*[Flag]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRow(row*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRow(row*[integer]*)
- SetFlag(flag*[Flag]*)
- SetRow(row*[integer]*, data*[Array of data]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)

- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Rigidwall constants

Name	Description
Rigidwall.CYLINDER	Rigidwall is *RIGIDWALL_GEOMETRIC_CYLINDER.
Rigidwall.FLAT	Rigidwall is *RIGIDWALL_GEOMETRIC_FLAT.
Rigidwall.PLANAR	Rigidwall is *RIGIDWALL_PLANAR.
Rigidwall.PRISM	Rigidwall is *RIGIDWALL_GEOMETRIC_PRISM.
Rigidwall.SPHERE	Rigidwall is *RIGIDWALL_GEOMETRIC_SPHERE.

Rigidwall properties

Name	Type	Description
birth	real	Birth time.
boxid	integer	Box for nodes.
d1	real	X component of vector defn.
d2	real	Y component of vector defn.
d3	real	Z component of vector defn.
death	real	Death time.
decaya	real	Friction decay const in local A dir.
decayb	real	Friction decay const in local B dir.
dfrica	real	Dynamic friction coeff in local A dir.
dfricb	real	Dynamic friction coeff in local B dir.
display	logical	DISPLAY flag.
e	real	Young’s modulus of rigidwall (for _DISPLAY option).
exists	logical	true if rigidwall exists, false if referred to but not defined. (read only)
finite	logical	Finite flag.
forces	logical	Forces flag.
fric	real	Friction coefficient.
heading	string	Rigidwall heading
id	logical	true if _ID option is set, false if not
include	integer	The Include file number that the rigidwall is in.
label	integer	Rigidwall number.
lcid	integer	Vel/disp vs time curve number.
lencyl	real	Length of cylinder.
lenl	real	Length of L edge.
lenm	real	Length of M edge.
lenp	real	Length of prism in -ve N.

mass	real	Mass of moving wall.
model	integer	The Model number that the rigidwall is in.
motion	logical	Motion flag.
moving	logical	Moving flag.
n1	integer	1st node for visualisation.
n2	integer	2nd node for visualisation.
n3	integer	3rd node for visualisation.
n4	integer	4th node for visualisation.
node1	integer	Node 1 for vector defn.
node2	integer	Node 2 for vector defn.
nsegs	integer	Number of subsections.
nsid	integer	Slave node set included in wall.
nsidex	integer	Slave node set exempted from wall.
offset	real	Offset for planar option.
opt	integer	Motion type.
ortho	logical	Ortho flag.
pid	integer	Part ID for display of geometric rigidwall (for _DISPLAY option).
pr	real	Poisson’s ratio of rigidwall (for _DISPLAY option).
radcyl	real	Radius of cylinder.
radsph	real	Radius of sphere.
ro	real	Density of rigidwall (for _DISPLAY option).
rwid	integer	Rigidwall number (identical to label).
rwksf	real	Stiffness scaling factor.
sfrica	real	Static friction coeff in local A dir.
sfricb	real	Static friction coeff in local B dir.
soft	integer	No. of cycles to zero relative velocity.
ssid	integer	Segment set number.
type	constant	The rigidwall type. Can be Rigidwall.FLAT, Rigidwall.PRISM, Rigidwall.CYLINDER, Rigidwall.SPHERE, Rigidwall.PLANAR,
v0	real	Initial velocity.
vx	real	X component of motion vector.
vy	real	Y component of motion vector.
vz	real	Z component of motion vector.
wvel	real	Velocity at which nodes weld to wall.
xh	real	Head X coord of outward normal.
xhev	real	Head X coord of edge I vector.
xt	real	Tail X coord of outward normal.
yh	real	Head Y coord of outward normal.
yhev	real	Head Y coord of edge I vector.

yt	real	Tail Y coord of outward normal.
zh	real	Head Z coord of outward normal.
zhev	real	Head Z coord of edge I vector.
zt	real	Tail Z coord of outward normal.

Detailed Description

The Rigidwall class allows you to create, modify, edit rigidwall cards. See the documentation below for more details.

Constructor

new Rigidwall(Model*[Model]*, type*[constant]*, nsid (optional)*[integer]*, rwid (optional)*[integer]*, heading (optional)*[string]*)

Description

Create a new Rigidwall object.

Arguments

Name	Type	Description
Model	Model	Model that Rigidwall will be created in
type	constant	Specify the type of rigidwall (Can be Rigidwall.FLAT, Rigidwall.PRISM, Rigidwall.CYLINDER, Rigidwall.SPHERE, Rigidwall.PLANAR)
nsid (optional)	integer	Node set number.
rwid (optional)	integer	Rigidwall number
heading (optional)	string	Title for the Rigidwall

Return type

Rigidwall object

Example

To create a new rigidwall 200 of type GEOMETRIC_SPHERE in model m using node set 100 having the title "test wall"

```
var r = new Rigidwall(m, Rigidwall.SPHERE, 200, 100, "test wall");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a rigidwall.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the rigidwall

Return type

No return value

Example

To associate comment c to the rigidwall r:

```
r.AssociateComment(c);
```

Blank()

Description

Blanks the rigidwall

Arguments

No arguments

Return type

No return value

Example

To blank rigidwall r:

```
r.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the rigidwalls in model m:

```
Rigidwall.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged rigidwalls will be blanked in

flag	Flag	Flag set on the rigidwalls that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the rigidwalls in model m flagged with f:

```
Rigidwall.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the rigidwall is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if rigidwall r is blanked:

```
if (r.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	**Type**	**Description**
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse rigidwall r:

```
r.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the rigidwall.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the rigidwall

Return type

No return value

Example

To clear flag f for rigidwall r:

```
r.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the rigidwall.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Rigidwall object

Example

To copy rigidwall r into rigidwall z:

```
var z = r.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a rigidwall.

Arguments

Name	Type	Description
Model	Model	Model that the rigidwall will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Rigidwall object (or null if not made)

Example

To start creating a rigidwall in model m:

```
var r = Rigidwall.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a rigidwall.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the rigidwall

Return type

No return value

Example

To detach comment c from the rigidwall r:

```
r.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit rigidwall r:

```
r.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for rigidwall. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for rigidwall r:

```
r.Error("My custom error");
```

FindNodesBehind(flag[*Flag*])

Description

Flags nodes that are behind a rigidwall

Arguments

Name	Type	Description
flag	Flag	Flag to be set on nodes behind rigidwall.

Return type

Number of nodes found

Example

To set flag f on nodes behind rigidwall w:

```
w.FlagNodesBehind(f);
```

First(Model[*Model*]) [static]

Description

Returns the first rigidwall in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first rigidwall in

Return type

Rigidwall object (or null if there are no rigidwalls in the model).

Example

To get the first rigidwall in model m:

```
var r = Rigidwall.First(m);
```

FirstFreeLabel(Model[*Model*], layer (optional)[*Include number*]) [static]

Description

Returns the first free rigidwall label in the model. Also see Rigidwall.LastFreeLabel(), Rigidwall.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free rigidwall label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Rigidwall label.

Example

To get the first free rigidwall label in model m:

```
var label = Rigidwall.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the rigidwalls in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be flagged in
flag	Flag	Flag to set on the rigidwalls

Return type

No return value

Example

To flag all of the rigidwalls with flag f in model m:

```
Rigidwall.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the rigidwall is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the rigidwall

Return type

true if flagged, false if not.

Example

To check if rigidwall r has flag f set on it:

```
if (r.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each rigidwall in the model.
Note that ForEach has been designed to make looping over rigidwalls as fast as possible and so has some limitations.
Firstly, a single temporary Rigidwall object is created and on each function call it is updated with the current rigidwall data. This means that you should not try to store the Rigidwall object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new rigidwalls inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls are in
func	function	Function to call for each rigidwall
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the rigidwalls in model m:

```
Rigidwall.ForEach(m, test);
function test(r)
{
// r is Rigidwall object
}
```

To call function test for all of the rigidwalls in model m with optional object:

```
var data = { x:0, y:0 };
Rigidwall.ForEach(m, test, data);
function test(r, extra)
{
// r is Rigidwall object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Rigidwall objects for all of the rigidwalls in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get rigidwalls from

Return type

Array of Rigidwall objects

Example

To make an array of Rigidwall objects for all of the rigidwalls in model m

```
var r = Rigidwall.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a rigidwall.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the rigidwall r:

```
var comm_array = r.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Rigidwall objects for all of the flagged rigidwalls in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get rigidwalls from
flag	Flag	Flag set on the rigidwalls that you want to retrieve

Return type

Array of Rigidwall objects

Example

To make an array of Rigidwall objects for all of the rigidwalls in model m flagged with f

```
var r = Rigidwall.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Rigidwall object for a rigidwall ID.

Arguments

Name	Type	Description
Model	Model	Model to find the rigidwall in
number	integer	number of the rigidwall you want the Rigidwall object for

Return type

Rigidwall object (or null if rigidwall does not exist).

Example

To get the Rigidwall object for rigidwall 100 in model m

```
var r = Rigidwall.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Rigidwall property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Rigidwall.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	rigidwall property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Rigidwall property r.example is a parameter:

```
Options.property_parameter_names = true;
if (r.GetParameter(r.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Rigidwall property r.example is a parameter by using the GetParameter method:

```
if (r.ViewParameters().GetParameter(r.example) ) do_something...
```

GetRow(row*[integer]*)

Description

Returns the data for an NSEGS card row in the rigidwall.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

An array of numbers containing the row variables VL and HEIGHT.

Example

To get the data for the 2nd row in rigidwall r:

```
var data = r.GetRow(1);
var vl = data[0];
var height = data[1];
```

Keyword()

Description

Returns the keyword for this Rigidwall (*RIGIDWALL). **Note that a carriage return is not added**. See also Rigidwall.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Rigidwall pm:

```
var key = r.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Rigidwall. **Note that a carriage return is not added**. See also Rigidwall.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Rigidwall pm:

```
var cards = r.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last rigidwall in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last rigidwall in

Return type

Rigidwall object (or null if there are no rigidwalls in the model).

Example

To get the last rigidwall in model m:

```
var r = Rigidwall.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free rigidwall label in the model. Also see Rigidwall.FirstFreeLabel(), Rigidwall.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free rigidwall label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Rigidwall label.

Example

To get the last free rigidwall label in model m:

```
var label = Rigidwall.LastFreeLabel(m);
```

Next()

Description

Returns the next rigidwall in the model.

Arguments

No arguments

Return type

Rigidwall object (or null if there are no more rigidwalls in the model).

Example

To get the rigidwall in model m after rigidwall r:

```
var r = r.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) rigidwall label in the model. Also see Rigidwall.FirstFreeLabel(), Rigidwall.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free rigidwall label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Rigidwall label.

Example

To get the next free rigidwall label in model m:

```
var label = Rigidwall.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a rigidwall.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only rigidwalls from that model can be picked. If the argument is a Flag then only rigidwalls that are flagged with *limit* can be selected. If omitted, or null, any rigidwalls from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Rigidwall object (or null if not picked)

Example

To pick a rigidwall from model m giving the prompt 'Pick rigidwall from screen':

```
var r = Rigidwall.Pick('Pick rigidwall from screen', m);
```

Previous()

Description

Returns the previous rigidwall in the model.

Arguments

No arguments

Return type

Rigidwall object (or null if there are no more rigidwalls in the model).

Example

To get the rigidwall in model m before rigidwall r:

```
var r = r.Previous();
```

RemoveRow(row*[integer]*)

Description

Removes an NSEGS card row in the *RIGIDWALL.

Arguments

Name	Type	Description
row	integer	The row you want to remove the data for. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second row of data for rigidwall r:

```
r.RemoveRow(1);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the rigidwalls in model m, from 1000000:

```
Rigidwall.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged rigidwalls will be renumbered in
flag	Flag	Flag set on the rigidwalls that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the rigidwalls in model m flagged with f, from 1000000:

```
Rigidwall.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select rigidwalls using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting rigidwalls
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only rigidwalls from that model can be selected. If the argument is a Flag then only rigidwalls that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any rigidwalls can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of rigidwalls selected or null if menu cancelled

Example

To select rigidwalls from model m, flagging those selected with flag f, giving the prompt 'Select rigidwalls':

```
Rigidwall.Select(f, 'Select rigidwalls', m);
```

To select rigidwalls, flagging those selected with flag f but limiting selection to rigidwalls flagged with flag l, giving the prompt 'Select rigidwalls':

```
Rigidwall.Select(f, 'Select rigidwalls', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the rigidwall.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the rigidwall

Return type

No return value

Example

To set flag f for rigidwall r:

```
r.SetFlag(f);
```

SetRow(row*[integer]*, data*[Array of data]*)

Description

Sets the data for an NSEGS card row in the *RIGIDWALL.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that row indices start at 0**.
data	Array of data	The data you want to set the row to

Return type

No return value.

Example

To set the second row of data for rigidwall r to be vl 10.0 and height 1.0:

```
var array = [10.0, 1.0];
r.SetRow(1, array);
```

To append a new row of data (using the same array of values):

```
r.SetRow(r.nsegs, array);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the rigidwall. The rigidwall will be sketched until you either call Rigidwall.Unsketch(), Rigidwall.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the rigidwall is sketched. If omitted redraw is true. If you want to sketch several rigidwalls and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch rigidwall r:

```
r.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged rigidwalls in the model. The rigidwalls will be sketched until you either call Rigidwall.Unsketch(), Rigidwall.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged rigidwalls will be sketched in

flag	Flag	Flag set on the rigidwalls that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the rigidwalls are sketched. If omitted redraw is true. If you want to sketch flagged rigidwalls several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all rigidwalls flagged with flag in model m:

```
Rigidwall.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing rigidwalls should be counted. If false or omitted referenced but undefined rigidwalls will also be included in the total.

Return type

number of rigidwalls

Example

To get the total number of rigidwalls in model m:

```
var total = Rigidwall.Total(m);
```

Unblank()

Description

Unblanks the rigidwall

Arguments

No arguments

Return type

No return value

Example

To unblank rigidwall r:

```
r.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the rigidwalls in model m:

```
Rigidwall.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged rigidwalls will be unblanked in
flag	Flag	Flag set on the rigidwalls that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the rigidwalls in model m flagged with f:

```
Rigidwall.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all rigidwalls will be unset in

flag	Flag	Flag to unset on the rigidwalls

Return type

No return value

Example

To unset the flag f on all the rigidwalls in model m:

```
Rigidwall.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the rigidwall.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the rigidwall is unsketched. If omitted redraw is true. If you want to unsketch several rigidwalls and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch rigidwall r:

```
r.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all rigidwalls.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the rigidwalls are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all rigidwalls in model m:

```
Rigidwall.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged rigidwalls in the model.

Arguments

Name	Type	Description
Model	Model	Model that all rigidwalls will be unsketched in
flag	Flag	Flag set on the rigidwalls that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the rigidwalls are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all rigidwalls flagged with flag in model m:

```
Rigidwall.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Rigidwall object.

Example

To check if Rigidwall property r.example is a parameter by using the Rigidwall.GetParameter() method:

```
if (r.ViewParameters().GetParameter(r.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for rigidwall. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for rigidwall r:

```
r.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this rigidwall.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for rigidwall r:

```
var xrefs = r.Xrefs();
```

---

toString()

Description

Creates a string containing the Rigidwall data in keyword format. Note that this contains the keyword header and the keyword cards. See also Rigidwall.Keyword() and Rigidwall.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Rigidwall pm in keyword format

```
var r = r.toString();
```

Section class

The Section class gives you access to section cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetBetaData(ipt*[integer]*)
- GetComments()
- GetLmcData(i*[integer]*)
- GetParameter(prop*[string]*)
- GetPointData(ipt*[integer]*)
- GetUserData(ipt*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetBetaData(ipt*[integer]*, beta*[real]*)
- SetFlag(flag*[Flag]*)
- SetLmcData(ipt*[integer]*, lmc*[real]*)

- SetPointData(ipt*[integer]*, nodeid*[integer]*, vecid*[integer]*, area*[real]*)
- SetUserData(ipt*[integer]*, xi*[real]*, eta*[real]*, zeta (SOLID) **or**
  wgt (SHELL)*[real]*, wgt (SOLID only)*[real]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Section constants

Name	Description
Section.ALE1D	Section ale1d type
Section.ALE2D	Section ale2d type
Section.BEAM	Section beam type
Section.DISCRETE	Section discrete type
Section.POINT_SOURCE	Section point source type
Section.SEATBELT	Section seatbelt type
Section.SHELL	Section shell type
Section.SOLID	Section solid type
Section.SPH	Section sph type
Section.TSHELL	Section thick shell type

Section properties

Name	Type	Description
aafac	real	ALE advection factor(SHELL, SOLID)
aet	integer	Ambient element type (ALE1D, ALE2D, SOLID)
afac	real	Smoothing weight factor - Simple average (SHELL, SOLID)
ale	logical	If _ALE option is set. Can be true or false (SHELL, SOLID)
aleform	integer	ALE formulation (ALE1D, ALE2D)
baselm	integer	Base element type for XFEM (SHELL)
bfac	real	Smoothing weight factor - Volume weighting (SHELL, SOLID)
cfac	real	Smoothing weight factor - Isoparametric (SHELL, SOLID)
cmid	integer	Cohesive material (SHELL, SOLID)
colour	Colour	The colour of the section
dfac	real	Smoothing weight factor - Equipotential (SHELL, SOLID)
domint	integer	Domain integration in XFEM (SHELL)
dr	real	PERI normalized horizon size (SOLID)
dx	real	Normalized dilation parameter of kernel function in X (SHELL, SOLID)
dy	real	Normalized dilation parameter of kernel function in Y (SHELL, SOLID)
dz	real	Normalized dilation parameter of kernel function in Z (SOLID)
efac	real	Smoothing weight factor - Equilibrium (SHELL)

efg	logical	If _EFG option is set. Can be true or false (SHELL, SOLID)
elform	integer	Element formulation (ALE1D, ALE2D, BEAM, SHELL, SOLID, TSHELL)
end	real	End time for smoothing (SHELL, SOLID)
exists	logical	true if section exists, false if referred to but not defined. (read only)
failcr	integer	Different failure criteria (SHELL)
fs	real	SPG Failure strain if IDAM = 1 (SOLID)
icomp	integer	Composite flag (SHELL, TSHELL)
idam	integer	SPG Option of damage mechanism (SOLID)
idila	integer	Normalized dilation parameter definition (SOLID)
idim	integer	Domain integration method (SOLID)
iebt	integer	Essential boundary condition treatment (SOLID)
ihgf	integer	Flag for using hourglass stabilization (SHELL, SOLID)
iloc	integer	Coordinate system option (SHELL)
include	integer	The Include file number that the section is in.
ispline	integer	EFG kernel function definition (SHELL, SOLID)
itaj	integer	Flag for setting up finite element matrices (SHELL, SOLID)
itb	integer	SPG Stabilization flag (SOLID)
ithelfm	integer	THERMAL shell formulation (SHELL)
iunf	integer	Flag for using nodal fibre vectors (SHELL)
kernel	integer	SPG kernel type approximation (SOLID)
label	integer or string	Section ID (all types) or character label. Also see the secid property which is an alternative name for this.
lmc	integer	Number of property parameters (SHELL, SOLID)
lprint	integer	Debug printout option (SHELL)
lscale	real	SPG length scale for displacement regularisation (SOLID)
model	integer	The Model number that the section is in.
nhsv	integer	Number of history variables (SHELL, SOLID)
nip	integer	Number of integration points (SHELL, SOLID, TSHELL)
nipp	integer	Number of in-plane integration points (SHELL)
nxdof	integer	Number of extra degrees of freedom per node (SHELL, SOLID)
peri	logical	If _PERI option is set. Can be true or false (SOLID)
propcr	integer	Not used (SHELL)
propt	real	Printout option (SHELL, TSHELL)
ptype	integer	PERI peridynamics formulation (SOLID)
qr	real	Quadrature rule (BEAM, SHELL, TSHELL)
secid	integer or string	Section ID (all types) or character label. Also see the label property which is an alternative name for this.
shrf	real	Shear correction factor (BEAM, SHELL, TSHELL)
smstep	integer	SPG Interval of timestep to conduction displ regularisation (SOLID)
spg	logical	If _SPG option is set. Can be true or false (SOLID)

start	real	Time imposed SPH approximation is activated (SPH) **or** Start time for smoothing (SHELL, SOLID)
stretch	real	SPG stretching parameter if IDAM = 1 (SOLID)
swtime	real	SPG Time to switch from updated Lagrangian to Eulerian kernel (SOLID)
thermal	logical	If _THERMAL option is set. Can be true or false (SHELL)
thick	real	Thickness (ALE1D, SEATBELT)
title	string	Section title (all types)
toldef	real	Deformation tolerance (SOLID)
transparency	integer	The transparency of the section (0-100) 0% is opaque, 100% is transparent.
type (read only)	constant	Section type. Can be Section.ALE1D, Section.ALE2D, Section.BEAM, Section.DISCRETE, Section.POINT_SOURCE, Section.SEATBELT, Section.SHELL, Section.SOLID, Section.SPH or Section.TSHELL
xfem	logical	If _THERMAL option is set. Can be true or false (SHELL)

Properties for BEAM

Name	Type	Description
a	real	Cross sectional area
aisc	logical	If _AISC option is set. Can be true or false
aisc_label	string	AISC section label
ca	real	Cable area
cid	integer	Coordinate system ID for orientation
cst	real	Cross section type
d1	real	Input parameter 1 for section type
d2	real	Input parameter 2 for section type
d3	real	Input parameter 3 for section type
d4	real	Input parameter 4 for section type
d5	real	Input parameter 5 for section type
d6	real	Input parameter 6 for section type
dofn1	real	Active degree of freedom at node 1
dofn2	real	Active degree of freedom at node 2
iner	real	Mass moment of inertia
iovpr	integer	Print flag for the elbow ovalization degrees of freedom (elform 14)
iprstr	integer	Flag for adding stress due to pressure into the material routine (elform 14)
irr	real	Irr
iss	real	Iss
ist	real	Ist
itoff	real	Option to specify torsional behaviour for spotwelds
itorm	real	Itorm
itt	real	Itt
iw	real	Warping constant
iwr	real	Warping constant

iyr	real	IYR integral
izr	real	IZR integral
j	real	torsional constant
nsloc	real	Location of s reference surface
nsm	real	Non structural mass per unit length
ntloc	real	Location of t reference surface
offset	real	Offset for cable
pr	real	Pressure inside elements (elform 14)
print	real	Output spot force resultants from spotwelds
rampt	real	Ramp up time for dynamic relaxation
rrcon	real	r rotational constraint
sa	real	Shear area
scoor	real	Location of triad for discrete beam
srcon	real	s rotational constraint
stress	real	Initial stress for dynamic relaxation
stype	string	Section type
trcon	real	t rotational constraint
ts1	real	s thickness or outer diameter at N1
ts2	real	s thickness or outer diameter at N2
tt1	real	t thickness or inner diameter at N1
tt2	real	t thickness or inner diameter at N2
vol	real	Volume of discrete beam
ys	real	s coordinate of shear centre of cross section
zs	real	t coordinate of shear centre of cross section

Properties for DISCRETE

Name	Type	Description
cdl	real	Deflection limit in compression
cl	real	Clearance
dro	integer	Displacement/rotation option
fd	real	Failure deflection
kd	real	Dynamic magnification factor
tdl	real	Deflection limit in tension
v0	real	Test velocity

Properties for POINT SOURCE

Name	Type	Description
lcidt	integer	Temperature loadcurve ID
lcidvel	integer	Inlet flow velocity loadcurve ID
lcidvolr	integer	Relative volume loadcurve ID
lcmdot1	integer	Mass flowrate loadcurve for gas 1

lcmdot2	integer	Mass flowrate loadcurve for gas 2
lcmdot3	integer	Mass flowrate loadcurve for gas 3
lcmdot4	integer	Mass flowrate loadcurve for gas 4
lcmdot5	integer	Mass flowrate loadcurve for gas 5
lcmdot6	integer	Mass flowrate loadcurve for gas 6
lcmdot7	integer	Mass flowrate loadcurve for gas 7
lcmdot8	integer	Mass flowrate loadcurve for gas 8
mixture	logical	If _MIXTURE option is set. Can be true or false
nidlc001	integer	1st node ID defining a local coordinate
nidlc002	integer	2nd node ID defining a local coordinate
nidlc003	integer	3rd node ID defining a local coordinate
points	integer	Number of point sources

Properties for SEATBELT

Name	Type	Description
area	real	Optional cross sectional area used in contact

Properties for SHELL

Name	Type	Description
edgset	integer	Edge node set
idof	real	Thickness field value
marea	real	Non structural mass per unit area
nloc	integer	Location of reference surface
setyp	integer	2D solid element type
t1	real	Thickness at Node 1
t2	real	Thickness at Node 2
t3	real	Thickness at Node 3
t4	real	Thickness at Node 4

Properties for SOLID

Name	Type	Description
cohoff	real	Relative location of cohesive layer (for cohesive solid elements 20 and 22)
ds	real	Displacement jump
ecut	real	Minimum distance to the node that a crack surface can cut to the edge
gaskett	real	Gasket thickness for converting elform 19-22 elements to gasket element
ibr	integer	Branching
iken	integer	approximation
ips	integer	Pressure smoothing/recovery
sf	real	Failure strain
stime	real	Time to switch from stabilized EFG to standard EFG formulation

Properties for SPH

Name	Type	Description
cslh	real	Smoothing length constant
death	real	Time imposed SPH approximation is stopped
ellipse	logical	If _ELLIPSE option is set (was _TENSOR pre R8). Can be true or false
hmax	real	Max smoothing length scale factor
hmin	real	Min smoothing length scale factor
hxcslh	real	Constant for smoothing length in X for tensor/ellipse case
hxini	real	Initial smoothing length in X for tensor/ellipse case
hycslh	real	Constant for smoothing length in Y for tensor/ellipse case
hyini	real	Initial smoothing length in Y for tensor/ellipse case
hzcslh	real	Constant for smoothing length in Z for tensor/ellipse case
hzini	real	Initial smoothing length in Z for tensor/ellipse case
iform	integer	SPH element formulation
interaction	logical	If _INTERACTION option is set. Can be true or false
sphini	real	Optional initial smoothing length
tensor	logical	If _TENSOR option is set (_ELLIPSE from R8 onwards). Can be true or false
user	logical	If _USER option is set. Can be true or false

Properties for TSHELL

Name	Type	Description
tshear	integer	Flag for transverse shear strain or stress distribution

Detailed Description

The Section class allows you to create, modify, edit and manipulate section cards. See the documentation below for more details.

Constructor

new Section(Model*[Model]*, secid*[integer or string]*, type*[constant]*, title (optional)*[string]*)

Description

Create a new Section object.

Arguments

Name	Type	Description
Model	Model	Model that section will be created in
secid	integer or string	Section number or character label
type	constant	Section type. Can be Section.BEAM, Section.DISCRETE, Section.POINT_SOURCE, Section.SEATBELT, Section.SHELL, Section.SOLID, Section.SPH or Section.TSHELL

title (optional)	string	Title for the section

Return type

Section object

Example

To create a new section, type shell, called 'Example' in model m with label 100:

```
var s = new Section(m, 100, Section.SHELL, 'Example');
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the section

Return type

No return value

Example

To associate comment c to the section s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the section

Arguments

No arguments

Return type

No return value

Example

To blank section s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sections in model m:

```
Section.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sections will be blanked in
flag	Flag	Flag set on the sections that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sections in model m flagged with f:

```
Section.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the section is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if section s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse section s:

```
s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the section.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the section

Return type

No return value

Example

To clear flag f for section s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the section.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Section object

Example

To copy section s into section z:

```
var z = s.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a section.

Arguments

Name	Type	Description
Model	Model	Model that the sect will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Section object (or null if not made)

Example

To start creating a section in model m:

```
var d = Section.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a section.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the section

Return type

No return value

Example

To detach comment c from the section s:

```
s.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit section s:

```
s.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for section s:

```
s.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for section.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the section colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the section.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing section s:

```
var colour = s.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first section in

Return type

Section object (or null if there are no sections in the model).

Example

To get the first section in model m:

```
var s = Section.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free section label in the model. Also see Section.LastFreeLabel(), Section.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Section label.

Example

To get the first free section label in model m:

```
var label = Section.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the sections in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be flagged in
flag	Flag	Flag to set on the sections

Return type

No return value

Example

To flag all of the sections with flag f in model m:

```
Section.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the section is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the section

Return type

true if flagged, false if not.

Example

To check if section s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each section in the model.
Note that ForEach has been designed to make looping over sections as fast as possible and so has some limitations.
Firstly, a single temporary Section object is created and on each function call it is updated with the current section data. This means that you should not try to store the Section object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new sections inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all sections are in
func	function	Function to call for each section
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the sections in model m:

```
Section.ForEach(m, test);
function test(s)
{
// s is Section object
}
```

To call function test for all of the sections in model m with optional object:

```
var data = { x:0, y:0 };
Section.ForEach(m, test, data);
function test(s, extra)
{
// s is Section object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Section objects for all of the sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sections from

Return type

Array of Section objects

Example

To make an array of Section objects for all of the sections in model m

```
var s = Section.GetAll(m);
```

GetBetaData(ipt*[integer]*)

Description

Returns the beta angle data for an integration point in *SECTION_SHELL or *SECTION_TSHELL.

Arguments

Name	Type	Description
ipt	integer	The integration point you want the data for. **Note that integration points start at 0, not 1**.

Return type

real

Example

To get the beta angle for the 3rd integration point for section shell s:

```
if (s.icomp && s.nip >= 3)
{
    var beta = s.GetBetaData(2);
}
```

GetComments()

Description

Extracts the comments associated to a section.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the section s:

```
var comm_array = s.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Section objects for all of the flagged sections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sections from
flag	Flag	Flag set on the sections that you want to retrieve

Return type

Array of Section objects

Example

To make an array of Section objects for all of the sections in model m flagged with f

```
var s = Section.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Section object for a section ID.

Arguments

Name	Type	Description
Model	Model	Model to find the section in
number	integer	number of the section you want the Section object for

Return type

Section object (or null if section does not exist).

Example

To get the Section object for section 100 in model m

```
var s = Section.GetFromID(m, 100);
```

GetLmcData(i*[integer]*)

Description

Returns the LMC property parameter for *SECTION_SHELL or *SECTION_SOLID.

Arguments

Name	Type	Description
i	integer	The point you want the parameter for. **Note that points start at 0, not 1**.

Return type

real

Example

To get the 3rd LMC parameter for section shell s:

```
if (s.lmc >= 3)
{
    var p = s.GetLmcData(2);
}
```

GetParameter(prop*[string]*)

Description

Checks if a Section property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Section.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	section property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Section property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Section property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

GetPointData(ipt*[integer]*)

Description

Returns the point data for a single point in *SECTION_POINT_SOURCE.

Arguments

Name	Type	Description
ipt	integer	The point you want the data for. **Note that integration points start at 0, not 1**.

Return type

An array of numbers containing the node id, vector id and orifice area.

Example

To get the data for the 3rd point for section point source s:

```
if (s.points >= 3)
{
    var pt_data = s.GetPointData(3);
}
```

GetUserData(ipt*[integer]*)

Description

Returns the user defined data for an integration point in *SECTION_SHELL and *SECTION_SOLID.

Arguments

Name	Type	Description
ipt	integer	The integration point you want the data for. **Note that integration points start at 0, not 1**.

Return type

An array containing the data (XI, ETA, WGT for *SECTION_SHELL, XI, ETA, ZETA, WGT for *SECTION_SOLID).

Example

To get the data for the 3rd integration point for section shell s:

```
if (s.nipp >= 3)
{
    var user_data = s.GetUserData(2);
}
```

Keyword()

Description

Returns the keyword for this section (*SECT, *SECT_SCALAR or *SECT_SCALAR_VALUE). **Note that a carriage return is not added**. See also Section.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for section s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the section. **Note that a carriage return is not added**. See also Section.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for section s:

```
var cards = n.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last section in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last section in

Return type

Section object (or null if there are no sections in the model).

Example

To get the last section in model m:

```
var s = Section.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free section label in the model. Also see Section.FirstFreeLabel(), Section.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free section label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Section label.

Example

To get the last free section label in model m:

```
var label = Section.LastFreeLabel(m);
```

Next()

Description

Returns the next section in the model.

Arguments

No arguments

Return type

Section object (or null if there are no more sections in the model).

Example

To get the section in model m after section s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) section label in the model. Also see Section.FirstFreeLabel(), Section.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free section label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Section label.

Example

To get the next free section label in model m:

```
var label = Section.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a section.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sections from that model can be picked. If the argument is a Flag then only sections that are flagged with *limit* can be selected. If omitted, or null, any sections from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Section object (or null if not picked)

Example

To pick a section from model m giving the prompt 'Pick section from screen':

```
var s = Section.Pick('Pick section from screen', m);
```

Previous()

Description

Returns the previous section in the model.

Arguments

No arguments

Return type

Section object (or null if there are no more sections in the model).

Example

To get the section in model m before section s:

```
var s = s.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be renumbered in

start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sections in model m, from 1000000:

```
Section.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sections will be renumbered in
flag	Flag	Flag set on the sections that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the sections in model m flagged with f, from 1000000:

```
Section.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select sections using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting sections
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sections from that model can be selected. If the argument is a Flag then only sections that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any sections can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of sections selected or null if menu cancelled

Example

To select sections from model m, flagging those selected with flag f, giving the prompt 'Select sections':

```
Section.Select(f, 'Select sections', m);
```

To select sections, flagging those selected with flag f but limiting selection to sections flagged with flag l, giving the prompt 'Select sections':

```
Section.Select(f, 'Select sections', l);
```

SetBetaData(ipt*[integer]*, beta*[real]*)

Description

Sets the beta angle for an integration point in *SECTION_SHELL or *SECTION_TSHELL.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to set the data for. **Note that integration points start at 0, not 1**.
beta	real	Beta angle for the integration point.

Return type

No return value.

Example

To set the beta angle for the 3rd integration point to 45, for section s:

```
s.SetBetaData(2, 45);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the section.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the section

Return type

No return value

Example

To set flag f for section s:

```
s.SetFlag(f);
```

SetLmcData(ipt*[integer]*, lmc*[real]*)

Description

Sets the lmc parameter for a point in *SECTION_SHELL or *SECTION_SOLID.

Arguments

Name	Type	Description
ipt	integer	The point you want to set the data for. **Note that points start at 0, not 1**.
lmc	real	Lmc parameter for the point.

Return type

No return value.

Example

To set the 3rd lmc point to 0.1, for section s:

```
s.SetLmcData(2, 0.1);
```

SetPointData(ipt*[integer]*, nodeid*[integer]*, vecid*[integer]*, area*[real]*)

Description

Sets the data for a single point in *SECTION_POINT_SOURCE.

Arguments

Name	Type	Description
ipt	integer	The point you want to set the data for. **Note that integration points start at 0, not 1**.
nodeid	integer	Node ID for the point.
vecid	integer	Vector ID for the point.
area	real	Orifice area for the point.

Return type

No return value.

Example

To set the data for the 3rd point to node 1, vector 10 and area 0.2, for section s:

```
s.SetPointData(2, 1, 10, 0.2);
```

SetUserData(ipt*[integer]*, xi*[real]*, eta*[real]*, zeta (SOLID) **or** wgt (SHELL)*[real]*, wgt (SOLID only)*[real]*)

Description

Sets the user defined data for an integration point in *SECTION_SHELL and *SECTION_SOLID.

Arguments

Name	Type	Description
ipt	integer	The integration point you want to set the data for. **Note that integration points start at 0, not 1**.
xi	real	First isoparametric coordinate.
eta	real	Second isoparametric coordinate.
zeta (SOLID) **or** wgt (SHELL)	real	Second isoparametric coordinate (SOLID) **or** Isoparametric weight (SHELL)
wgt (SOLID only)	real	Isoparametric weight (SOLID)

Return type

No return value.

Example

To set the user data for the 3rd integration point to xi 0.5, eta 0.5, zeta -0.5, wgt 0.125, for section solid s:

```
s.SetUserData(2, 0.5, 0.5, -0.5, 0.125);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the section. The section will be sketched until you either call Section.Unsketch(), Section.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the section is sketched. If omitted redraw is true. If you want to sketch several sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch section s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged sections in the model. The sections will be sketched until you either call Section.Unsketch(), Section.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sections will be sketched in
flag	Flag	Flag set on the sections that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the sections are sketched. If omitted redraw is true. If you want to sketch flagged sections several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all sections flagged with flag in model m:

```
Section.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of sections in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing sections should be counted. If false or omitted referenced but undefined sections will also be included in the total.

Return type

number of sections

Example

To get the total number of sections in model m:

```
var total = Section.Total(m);
```

Unblank()

Description

Unblanks the section

Arguments

No arguments

Return type

No return value

Example

To unblank section s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sections in model m:

```
Section.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged sections will be unblanked in
flag	Flag	Flag set on the sections that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sections in model m flagged with f:

```
Section.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all sections will be unset in
flag	Flag	Flag to unset on the sections

Return type

No return value

Example

To unset the flag f on all the sections in model m:

```
Section.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the section.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the section is unsketched. If omitted redraw is true. If you want to unsketch several sections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch section s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all sections.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sections in model m:

```
Section.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged sections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sections will be unsketched in
flag	Flag	Flag set on the sections that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the sections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sections flagged with flag in model m:

```
Section.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Section object.

Example

To check if Section property s.example is a parameter by using the Section.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for section. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for section s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this section.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for section s:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the section data in keyword format. Note that this contains the keyword header and the keyword cards. See also Section.Keyword() and Section.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for section s in keyword format

```
var str = s.toString();
```

SensorControl class

The SensorControl class gives you access to *SENSOR_CONTROL keyword in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

SensorControl properties

Name	Type	Description
cntlid	integer	SensorControl number. The label property is an alternative name for this.

estyp	string	Element Set Type to be controlled. Can be "BEAM", "DISC", "SHELL", "SOLID", "TSHELL".
exists	logical	true if *SENSOR_CONTROL exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the *SENSOR_CONTROL is in.
initstt	string	Initial status. Can be "On" or "Off".
label	integer	SensorControl number. The cntlid property is an alternative name for this.
model	integer	The Model number that the *SENSOR_CONTROL is in.
nrep	integer	Number of repeat of cycle of switches.
swit1	integer	ID of 1st switch.
swit2	integer	ID of 2nd switch.
swit3	integer	ID of 3rd switch.
swit4	integer	ID of 4th switch.
swit5	integer	ID of 5th switch.
swit6	integer	ID of 6th switch.
swit7	integer	ID of 7th switch.
timeoff	integer	Flag for offset of time in curve.
type	string	Entity to be controlled. Can be "AIRBAG", "BAGVENTPOP", "BELTPRET", "BELTRETRA", "BELTSLIP", "CONTACT", "CONTACT2D", "CNRB", "DEF2RIG", "DISC-ELE", "DISC-ELES", "ELESET", "FUNCTION", "JOINT", "JOINTSTIFF", "LOADTHM", "M PRESSURE", "RWALL", "SPC", "SPOTWELD".
typeid	integer	ID of entity to be controlled if type is not FUNCTION or input value for FUNCTION.

Detailed Description

The SensorControl class allows you to create, modify, edit and manipulate *SENSOR_CONTROL. See the documentation below for more details.

Constructor

new SensorControl(Model*[Model]*, Sensor control ID*[integer]*, Type*[string]*, Type ID (optional)*[integer]*, estyp (optional)*[string]*)

Description

Create a new SensorControl object.

Arguments

Name	Type	Description
Model	Model	Model that *SENSOR_CONTROL will be created in
Sensor control ID	integer	SensorControl id.
Type	string	Entity type to be controlled. Can be "AIRBAG", "BAGVENTPOP", "BELTPRET", "BELTRETRA", "BELTSLIP", "CONTACT", "CONTACT2D", "DEF2RIG", "DISC-ELE", "DISC-ELES", "ELESET", "FUNCTION", "JOINT", "JOINTSTIFF", "M PRESSURE", "RWALL", "SPC", "SPOTWELD".
Type ID (optional)	integer	ID of entity to be controlled if type is not FUNCTION or input value for FUNCTION.
estyp (optional)	string	Element Set Type to be controlled. Can be "BEAM", "DISC", "SHELL", "SOLID", "TSHELL". **Required only if Type argument is "ELESET"**.

Return type

SensorControl object

Example

To create a new *SENSOR_CONTROL in model m with label 100 and type JOINT:

```
var sc = new SensorControl(m, 100, "JOINT");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a *SENSOR_CONTROL.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the *SENSOR_CONTROL

Return type

No return value

Example

To associate comment c to the *SENSOR_CONTROL sc:

```
sc.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse *SENSOR_CONTROL sc:

```
sc.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the *SENSOR_CONTROL.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the *SENSOR_CONTROL

Return type

No return value

Example

To clear flag f for *SENSOR_CONTROL sc:

```
sc.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the *SENSOR_CONTROL.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

SensorControl object

Example

To copy *SENSOR_CONTROL sc into *SENSOR_CONTROL z:

```
var z = sc.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a *SENSOR_CONTROL.

Arguments

Name	Type	Description
Model	Model	Model that the *SENSOR_CONTROL will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

SensorControl object (or null if not made)

Example

To start creating a *SENSOR_CONTROL in model m:

```
var sc = SensorControl.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a *SENSOR_CONTROL.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the *SENSOR_CONTROL

Return type

No return value

Example

To detach comment c from the *SENSOR_CONTROL sc:

```
sc.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit *SENSOR_CONTROL sc:

```
sc.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for *SENSOR_CONTROL. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for *SENSOR_CONTROL sc:

```
sc.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first *SENSOR_CONTROL in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first *SENSOR_CONTROL in

Return type

SensorControl object (or null if there are no *SENSOR_CONTROLs in the model).

Example

To get the first *SENSOR_CONTROL in model m:

```
var sc = SensorControl.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free *SENSOR_CONTROL label in the model. Also see SensorControl.LastFreeLabel(), SensorControl.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free *SENSOR_CONTROL label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

SensorControl label.

Example

To get the first free *SENSOR_CONTROL label in model m:

```
var label = SensorControl.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the *SENSOR_CONTROLs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_CONTROLs will be flagged in
flag	Flag	Flag to set on the *SENSOR_CONTROLs

Return type

No return value

Example

To flag all of the *SENSOR_CONTROLs with flag f in model m:

```
SensorControl.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the *SENSOR_CONTROL is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the *SENSOR_CONTROL

Return type

true if flagged, false if not.

Example

To check if *SENSOR_CONTROL sc has flag f set on it:

```
if (sc.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each *SENSOR_CONTROL in the model.

Note that ForEach has been designed to make looping over *SENSOR_CONTROLs as fast as possible and so has some limitations.
Firstly, a single temporary SensorControl object is created and on each function call it is updated with the current *SENSOR_CONTROL data. This means that you should not try to store the SensorControl object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new *SENSOR_CONTROLs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_CONTROLs are in
func	function	Function to call for each *SENSOR_CONTROL
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the *SENSOR_CONTROLs in model m:

```
SensorControl.ForEach(m, test);
function test(sc)
{
// sc is SensorControl object
}
```

To call function test for all of the *SENSOR_CONTROLs in model m with optional object:

```
var data = { x:0, y:0 };
SensorControl.ForEach(m, test, data);
function test(sc, extra)
{
// sc is SensorControl object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of SensorControl objects for all of the *SENSOR_CONTROLs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_CONTROLs from

Return type

Array of SensorControl objects

Example

To make an array of SensorControl objects for all of the *SENSOR_CONTROLs in model m

```
var sc = SensorControl.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a *SENSOR_CONTROL.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the *SENSOR_CONTROL sc:

```
var comm_array = sc.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of SensorControl objects for all of the flagged *SENSOR_CONTROLs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_CONTROLs from
flag	Flag	Flag set on the *SENSOR_CONTROLs that you want to retrieve

Return type

Array of SensorControl objects

Example

To make an array of SensorControl objects for all of the *SENSOR_CONTROLs in model m flagged with f

```
var sc = SensorControl.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the SensorControl object for a *SENSOR_CONTROL ID.

Arguments

Name	Type	Description
Model	Model	Model to find the *SENSOR_CONTROL in
number	integer	number of the *SENSOR_CONTROL you want the SensorControl object for

Return type

SensorControl object (or null if *SENSOR_CONTROL does not exist).

Example

To get the SensorControl object for *SENSOR_CONTROL 100 in model m

```
var sc = SensorControl.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a SensorControl property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the SensorControl.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	*SENSOR_CONTROL property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if SensorControl property sc.example is a parameter:

```
Options.property_parameter_names = true;
if (sc.GetParameter(sc.example) ) do_something...
Options.property_parameter_names = false;
```

To check if SensorControl property sc.example is a parameter by using the GetParameter method:

```
if (sc.ViewParameters().GetParameter(sc.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this *SENSOR_CONTROL. **Note that a carriage return is not added**. See also SensorControl.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for SensorControl sc:

```
var key = sc.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the *SENSOR_CONTROL. **Note that a carriage return is not added**. See also SensorControl.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for sensor control sc:

```
var cards = sc.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last *SENSOR_CONTROL in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last *SENSOR_CONTROL in

Return type

SensorControl object (or null if there are no *SENSOR_CONTROLs in the model).

Example

To get the last *SENSOR_CONTROL in model m:

```
var sc = SensorControl.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free *SENSOR_CONTROL label in the model. Also see SensorControl.FirstFreeLabel(), SensorControl.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free *SENSOR_CONTROL label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

SensorControl label.

Example

To get the last free *SENSOR_CONTROL label in model m:

```
var label = SensorControl.LastFreeLabel(m);
```

Next()

Description

Returns the next *SENSOR_CONTROL in the model.

Arguments

No arguments

Return type

SensorControl object (or null if there are no more *SENSOR_CONTROLs in the model).

Example

To get the *SENSOR_CONTROL in model m after *SENSOR_CONTROL sc:

```
var sc = sc.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) *SENSOR_CONTROL label in the model. Also see SensorControl.FirstFreeLabel(), SensorControl.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free *SENSOR_CONTROL label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

SensorControl label.

Example

To get the next free *SENSOR_CONTROL label in model m:

```
var label = SensorControl.NextFreeLabel(m);
```

Previous()

Description

Returns the previous *SENSOR_CONTROL in the model.

Arguments

No arguments

Return type

SensorControl object (or null if there are no more *SENSOR_CONTROLs in the model).

Example

To get the *SENSOR_CONTROL in model m before *SENSOR_CONTROL sc:

```
var sc = sc.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the *SENSOR_CONTROLs in the model.

Arguments

Name	Type	Description

Model	Model	Model that all *SENSOR_CONTROLs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_CONTROLs in model m, from 1000000:

```
SensorControl.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged *SENSOR_CONTROLs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *SENSOR_CONTROLs will be renumbered in
flag	Flag	Flag set on the *SENSOR_CONTROLs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_CONTROLs in model m flagged with f, from 1000000:

```
SensorControl.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select *SENSOR_CONTROLs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting *SENSOR_CONTROLs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only *SENSOR_CONTROLs from that model can be selected. If the argument is a Flag then only *SENSOR_CONTROLs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any *SENSOR_CONTROLs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of *SENSOR_CONTROLs selected or null if menu cancelled

Example

To select *SENSOR_CONTROLs from model m, flagging those selected with flag f, giving the prompt 'Select *SENSOR_CONTROLs':

```
SensorControl.Select(f, 'Select *SENSOR_CONTROLs', m);
```

To select *SENSOR_CONTROLs, flagging those selected with flag f but limiting selection to *SENSOR_CONTROLs flagged with flag l, giving the prompt 'Select *SENSOR_CONTROLs':

```
SensorControl.Select(f, 'Select *SENSOR_CONTROLs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the *SENSOR_CONTROL.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the *SENSOR_CONTROL

Return type

No return value

Example

To set flag f for *SENSOR_CONTROL sc:

```
sc.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the *SENSOR_CONTROL. The *SENSOR_CONTROL will be sketched until you either call SensorControl.Unsketch(), SensorControl.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the *SENSOR_CONTROL is sketched. If omitted redraw is true. If you want to sketch several *SENSOR_CONTROLs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch *SENSOR_CONTROL sc:

```
sc.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged *SENSOR_CONTROLs in the model. The *SENSOR_CONTROLs will be sketched until you either call SensorControl.Unsketch(), SensorControl.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *SENSOR_CONTROLs will be sketched in
flag	Flag	Flag set on the *SENSOR_CONTROLs that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the *SENSOR_CONTROLs are sketched. If omitted redraw is true. If you want to sketch flagged *SENSOR_CONTROLs several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all *SENSOR_CONTROLs flagged with flag in model m:

```
SensorControl.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of *SENSOR_CONTROLs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing *SENSOR_CONTROLs should be counted. If false or omitted referenced but undefined *SENSOR_CONTROLs will also be included in the total.

Return type

number of *SENSOR_CONTROLs

Example

To get the total number of *SENSOR_CONTROLs in model m:

```
var total = SensorControl.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the *SENSOR_CONTROLs in the model.

Arguments

Name	Type	Description

Model	Model	Model that the defined flag for all *SENSOR_CONTROLs will be unset in
flag	Flag	Flag to unset on the *SENSOR_CONTROLs

Return type

No return value

Example

To unset the flag f on all the *SENSOR_CONTROLs in model m:

```
SensorControl.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the *SENSOR_CONTROL.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the *SENSOR_CONTROL is unsketched. If omitted redraw is true. If you want to unsketch several *SENSOR_CONTROLs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch *SENSOR_CONTROL sc:

```
sc.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all *SENSOR_CONTROLs.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_CONTROLs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the *SENSOR_CONTROLs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all *SENSOR_CONTROLs in model m:

```
SensorControl.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged *SENSOR_CONTROLs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_CONTROLs will be unsketched in
flag	Flag	Flag set on the *SENSOR_CONTROLs that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the *SENSOR_CONTROLs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all *SENSOR_CONTROLs flagged with flag in model m:

```
SensorControl.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

SensorControl object.

Example

To check if SensorControl property sc.example is a parameter by using the SensorControl.GetParameter() method:

```
if (sc.ViewParameters().GetParameter(sc.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for *SENSOR_CONTROL. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give

details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for *SENSOR_CONTROL sc:

```
sc.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this *SENSOR_CONTROL.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for *SENSOR_CONTROL sc:

```
var xrefs = sc.Xrefs();
```

---

toString()

Description

Creates a string containing the sensor control data in keyword format. Note that this contains the keyword header and the keyword cards. See also SensorControl.Keyword() and SensorControl.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for sensor control sc in keyword format

```
var str = sc.toString();
```

SensorDefine class

The SensorDefine class gives you access to *SENSOR_DEFINE keyword in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

SensorDefine constants

Name	Description
SensorDefine.DEFINE_CALC_MATH	Sensor define is *SENSOR_DEFINE_CALC_MATH.
SensorDefine.DEFINE_ELEMENT	Sensor define is *SENSOR_DEFINE_ELEMENT.
SensorDefine.DEFINE_ELEMENT_SET	Sensor define is *SENSOR_DEFINE_ELEMENT_SET.
SensorDefine.DEFINE_FORCE	Sensor define is *SENSOR_DEFINE_FORCE.

SensorDefine.DEFINE_FUNCTION	Sensor define is *SENSOR_DEFINE_FUNCTION.
SensorDefine.DEFINE_MISC	Sensor define is *SENSOR_DEFINE_MISC.
SensorDefine.DEFINE_NODE	Sensor define is *SENSOR_DEFINE_NODE.
SensorDefine.DEFINE_NODE_SET	Sensor define is *SENSOR_DEFINE_NODE_SET.

SensorDefine properties

Name	Type	Description
calc	string	Mathematical calculation. Can be "ABSSUM", "MIN", "MAX", "MAXMAG", "MINMAG", "MULTIPLY", "SQRE", "SQRTSQRE", "SQRT", "SUMABS", "SUM" .
comp	string	Component type. Can be "XX", "YY", "ZZ", "XY", "YZ", "ZX", "AXIAL", "SHEARS", "SHEART".
crd	integer	Optional coordinate system.
ctype	string	Sensor type or Output component type. Can be "STRAIN", "STRESS", "FORCE", "MOMENT", "DLEN" or "FAIL" for SensorDefine.DEFINE_ELEMENT or SensorDefine.DEFINE_ELEMENT_SET and "ACC", "VEL", "COORD" or "TEMP" for SensorDefine.DEFINE_NODE or SensorDefine.DEFINE_NODE_SET
elemid	integer	Element ID or element set ID when option_SET is active.
etype	string	Element type. Can be "BEAM", "SHELL", "SOLID", "DISC-ELE", "SEATBELT" or "TSHELL".
exists	logical	true if *SENSOR_DEFINE exists, false if referred to but not defined. (read only)
ftype	string	Force type. Can be "AIRBAG", "CONTACT", "CONTACT2D", "CPM", "JOINT", "JOINTSTIF", "PRESC-MOT", "RWALL", "SPC", "SPOTWELD", "X-SECTION".
func	integer	Function ID.
func_sens1	integer	1st Sensor ID if positive or number of sensor ID if negative.
func_sens10	integer	10th Sensor ID.
func_sens11	integer	11th Sensor ID.
func_sens12	integer	12th Sensor ID.
func_sens13	integer	13th Sensor ID.
func_sens14	integer	14th Sensor ID.
func_sens15	integer	15th Sensor ID.
func_sens16	integer	16th Sensor ID.
func_sens2	integer	2nd Sensor ID.
func_sens3	integer	3rd Sensor ID.
func_sens4	integer	4th Sensor ID.
func_sens5	integer	5th Sensor ID.

func_ sens6	integer	6th Sensor ID.
func_ sens7	integer	7th Sensor ID.
func_ sens8	integer	8th Sensor ID.
func_ sens9	integer	9th Sensor ID.
i0	string	I0. Can be ’KINETIC", "INTERNAL", "ERODEKE", or "ERODEIE" when MTYPE = "MATSUM" or "SOLID", "SHELL", "TSHELL", "BEAM", or "DISC" when MTYPE = "NFAILE" or "TEMP", or "VOL" when MTYPE = "CVBAG".
i1	integer/string	I1. Applicable for "ANGLE", "BNDOUT", "CVBAG", "CURVE", "MATSUM" or "NFAILE".
i2	integer/string	I2. Applicable only for when MTYPE = "ANGLE".
i3	integer/string	I3. Applicable only for MTYPE = "ANGLE".
i4	integer/string	I4. Applicable only for MTYPE = "ANGLE".
i5	string	I5.
include	integer	The Include file number that the *SENSOR_DEFINE is in.
label	integer	SensorDefine number. The sensid property is an alternative name for this.
layer	integer/string	Layer of integration. Can be "BOT", "TOP" or "i" to monitor the stress of the ith integration point when ctype = "STRESS".
model	integer	The Model number that the *SENSOR_DEFINE is in.
mtype	string	Entity to be traced. Can be "ANGLE", "BNDOUT", "CURVE", "CVBAG", ’MATSUM’, "NFAILE", "RETRACTOR", "RIGIDBODY" .
node1	integer	Node or Node set ID based on option SET for an accelerometer sensor.
node2	integer	Node ID for an accelerometer sensor.
option	constant	SENSOR_DEFINE suffix. Can be SensorDefine.DEFINE_CALC_MATH, SensorDefine.DEFINE_ELEMENT, SensorDefine.DEFINE_ELEMENT_SET, SensorDefine.DEFINE_FORCE, SensorDefine.DEFINE_MISC, SensorDefine.DEFINE_ NODE, SensorDefine.DEFINE_NODE_SET or SensorDefine.DEFINE_FUNCTION.
pwr	real	Power (Optional parameters).
sens1	integer	1st Sensor ID.
sens2	integer	2nd Sensor ID.
sens3	integer	3rd Sensor ID.
sens4	integer	4th Sensor ID.
sens5	integer	5th Sensor ID.
sens6	integer	6th Sensor ID.
sensid	integer	SensorDefine number. The label property is an alternative name for this.
setopt	string	Option to process set of data when SET option is specified. Can be "AVG", "MAX", "MIN" or "SUM".
sf	real	Scale factor (Optional parameters).
typeid	integer	ID defined in the associated KEYWORD command.
vid	integer/string	Vector along which the forces is measured. Can be "X", "Y", "Z", "XMOMENT", "YMOMENT", "ZMOMENT" or vector ID n in coordinate system CRD for SensorDefine.DEFINE_FORCE or ID of vector along which the nodal values for SensorDefine.DEFINE_NODE and SensorDefine.DEFINE_NODE_SET.

Detailed Description

The SensorDefine class allows you to create, modify, edit and manipulate *SENSOR_DEFINE. See the documentation below for more details.

Constructor

new SensorDefine(Option*[constant]*, Model*[Model]*, Define ID*[integer]*, Type or Entity 1*[string/label]*, Entity 2*[label]*)

Description

Create a new SensorDefine object.

Arguments

Name	Type	Description
Option	constant	SENSOR_DEFINE suffix. Can be SensorDefine.DEFINE_CALC_MATH, SensorDefine.DEFINE_ELEMENT, SensorDefine.DEFINE_ELEMENT_SET, SensorDefine.DEFINE_FORCE, SensorDefine.DEFINE_MISC, SensorDefine.DEFINE_NODE, SensorDefine.DEFINE_NODE_SET or SensorDefine.DEFINE_FUNCTION.
Model	Model	Model that *SENSOR_DEFINE will be created in
Define ID	integer	SensorDefine id.
Type or Entity 1	string/label	For SensorDefine.DEFINE_NODE, SensorDefine.DEFINE_NODE_SET option it is Node ID or NODE set ID respectively, For SensorDefine.DEFINE_FUNCTION option it is DEFINE_FUNCTION ID, For SensorDefine.DEFINE_CALC_MATH option it is Calc string, For SensorDefine.DEFINE_ELEMENT and SensorDefine.DEFINE_ELEMENT_SET option it is Etype string, For SensorDefine.DEFINE_FORCE option it is Ftype string, For SensorDefine.DEFINE_MISC option it is Mtype string.
Entity 2	label	Applicable only for SensorDefine.DEFINE_NODE, SensorDefine.DEFINE_NODE_SET, SensorDefine.DEFINE_CALC_MATH, SensorDefine.DEFINE_ELEMENT, SensorDefine.DEFINE_ELEMENT_SET or SensorDefine.DEFINE_FORCE. It is NODE or NODE set ID for SensorDefine.DEFINE_NODE or SensorDefine.DEFINE_NODE_SET respectively, Sensor Define ID for SensorDefine.DEFINE_CALC_MATH, Element ID or Element set ID for SensorDefine.DEFINE_ELEMENT or SensorDefine.DEFINE_ELEMENT_SET respectively or Type ID for SensorDefine.DEFINE_FORCE.

Return type

SensorDefine object

Example

To create a new *SENSOR_DEFINE_CALC-MATH in model m with label 100 with CALC option as MAX and SENS1 as -2:

```
var sd1 = new SensorDefine(SensorDefine.DEFINE_CALC_MATH, m, 100, "MAX", -2);
```

To create a new *SENSOR_DEFINE_MISC in model m with label 10 with MTYPE option as ANGLE:

```
var sd2 = new SensorDefine(SensorDefine.DEFINE_MISC, m, 10, "ANGLE");
```

To create a new *SENSOR_DEFINE_NODE in model m with label 11 with NODE1 and NODE2 as 5 and 6:

```
var sd3 = new SensorDefine(SensorDefine.DEFINE_NODE, m, 11, 5, 6);
```

To create a new *SENSOR_DEFINE_FUNCTION in model m with label 12 and FUNCTION ID as 6:

```
var sd4 = new SensorDefine(SensorDefine.DEFINE_FUNCTION, m, 12, 6);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a *SENSOR_DEFINE.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the *SENSOR_DEFINE

Return type

No return value

Example

To associate comment c to the *SENSOR_DEFINE sd:

```
sd.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse *SENSOR_DEFINE sd:

```
sd.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the *SENSOR_DEFINE.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the *SENSOR_DEFINE

Return type

No return value

Example

To clear flag f for *SENSOR_DEFINE sd:

```
sd.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the *SENSOR_DEFINE.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

SensorDefine object

Example

To copy *SENSOR_DEFINE sd into *SENSOR_DEFINE z:

```
var z = sd.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a *SENSOR_DEFINE.

Arguments

Name	Type	Description
Model	Model	Model that the *SENSOR_DEFINE will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

SensorDefine object (or null if not made)

Example

To start creating a *SENSOR_DEFINE in model m:

```
var sd = SensorDefine.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a *SENSOR_DEFINE.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the *SENSOR_DEFINE

Return type

No return value

Example

To detach comment c from the *SENSOR_DEFINE sd:

```
sd.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit *SENSOR_DEFINE sd:

```
sd.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for *SENSOR_DEFINE. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for *SENSOR_DEFINE sd:

```
sd.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first *SENSOR_DEFINE in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first *SENSOR_DEFINE in

Return type

SensorDefine object (or null if there are no *SENSOR_DEFINEs in the model).

Example

To get the first *SENSOR_DEFINE in model m:

```
var sd = SensorDefine.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free *SENSOR_DEFINE label in the model. Also see SensorDefine.LastFreeLabel(), SensorDefine.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free *SENSOR_DEFINE label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

SensorDefine label.

Example

To get the first free *SENSOR_DEFINE label in model m:

```
var label = SensorDefine.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the *SENSOR_DEFINEs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_DEFINEs will be flagged in
flag	Flag	Flag to set on the *SENSOR_DEFINEs

Return type

No return value

Example

To flag all of the *SENSOR_DEFINEs with flag f in model m:

```
SensorDefine.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the *SENSOR_DEFINE is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the *SENSOR_DEFINE

Return type

true if flagged, false if not.

Example

To check if *SENSOR_DEFINE sd has flag f set on it:

```
if (sd.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each *SENSOR_DEFINE in the model.
Note that ForEach has been designed to make looping over *SENSOR_DEFINEs as fast as possible and so has some limitations.
Firstly, a single temporary SensorDefine object is created and on each function call it is updated with the current *SENSOR_DEFINE data. This means that you should not try to store the SensorDefine object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new *SENSOR_DEFINEs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_DEFINEs are in
func	function	Function to call for each *SENSOR_DEFINE
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the *SENSOR_DEFINEs in model m:

```
SensorDefine.ForEach(m, test);
function test(sd)
{
// sd is SensorDefine object
}
```

To call function test for all of the *SENSOR_DEFINEs in model m with optional object:

```
var data = { x:0, y:0 };
SensorDefine.ForEach(m, test, data);
function test(sd, extra)
{
// sd is SensorDefine object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of SensorDefine objects for all of the *SENSOR_DEFINEs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_DEFINEs from

Return type

Array of SensorDefine objects

Example

To make an array of SensorDefine objects for all of the *SENSOR_DEFINEs in model m

```
var sd = SensorDefine.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a *SENSOR_DEFINE.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the *SENSOR_DEFINE sd:

```
var comm_array = sd.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of SensorDefine objects for all of the flagged *SENSOR_DEFINEs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_DEFINEs from
flag	Flag	Flag set on the *SENSOR_DEFINEs that you want to retrieve

Return type

Array of SensorDefine objects

Example

To make an array of SensorDefine objects for all of the *SENSOR_DEFINEs in model m flagged with f

```
var sd = SensorDefine.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the SensorDefine object for a *SENSOR_DEFINE ID.

Arguments

Name	Type	Description
Model	Model	Model to find the *SENSOR_DEFINE in
number	integer	number of the *SENSOR_DEFINE you want the SensorDefine object for

Return type

SensorDefine object (or null if *SENSOR_DEFINE does not exist).

Example

To get the SensorDefine object for *SENSOR_DEFINE 100 in model m

```
var sd = SensorDefine.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a SensorDefine property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the SensorDefine.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description

prop	string	*SENSOR_DEFINE property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if SensorDefine property sd.example is a parameter:

```
Options.property_parameter_names = true;
if (sd.GetParameter(sd.example) ) do_something...
Options.property_parameter_names = false;
```

To check if SensorDefine property sd.example is a parameter by using the GetParameter method:

```
if (sd.ViewParameters().GetParameter(sd.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this *SENSOR_DEFINE. **Note that a carriage return is not added**. See also SensorDefine.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for SensorDefine sd:

```
var key = sd.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the *SENSOR_DEFINE. **Note that a carriage return is not added**. See also SensorDefine.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for sensor define sd:

```
var cards = sd.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last *SENSOR_DEFINE in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last *SENSOR_DEFINE in

Return type

SensorDefine object (or null if there are no *SENSOR_DEFINEs in the model).

Example

To get the last *SENSOR_DEFINE in model m:

```
var sd = SensorDefine.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free *SENSOR_DEFINE label in the model. Also see SensorDefine.FirstFreeLabel(), SensorDefine.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free *SENSOR_DEFINE label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

SensorDefine label.

Example

To get the last free *SENSOR_DEFINE label in model m:

```
var label = SensorDefine.LastFreeLabel(m);
```

Next()

Description

Returns the next *SENSOR_DEFINE in the model.

Arguments

No arguments

Return type

SensorDefine object (or null if there are no more *SENSOR_DEFINEs in the model).

Example

To get the *SENSOR_DEFINE in model m after *SENSOR_DEFINE sd:

```
var sd = sd.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) *SENSOR_DEFINE label in the model. Also see SensorDefine.FirstFreeLabel(), SensorDefine.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free *SENSOR_DEFINE label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

SensorDefine label.

Example

To get the next free *SENSOR_DEFINE label in model m:

```
var label = SensorDefine.NextFreeLabel(m);
```

Previous()

Description

Returns the previous *SENSOR_DEFINE in the model.

Arguments

No arguments

Return type

SensorDefine object (or null if there are no more *SENSOR_DEFINEs in the model).

Example

To get the *SENSOR_DEFINE in model m before *SENSOR_DEFINE sd:

```
var sd = sd.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the *SENSOR_DEFINEs in the model.

Arguments

Name	Type	Description

Model	Model	Model that all *SENSOR_DEFINEs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_DEFINEs in model m, from 1000000:

```
SensorDefine.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged *SENSOR_DEFINEs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *SENSOR_DEFINEs will be renumbered in
flag	Flag	Flag set on the *SENSOR_DEFINEs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_DEFINEs in model m flagged with f, from 1000000:

```
SensorDefine.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select *SENSOR_DEFINEs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting *SENSOR_DEFINEs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only *SENSOR_DEFINEs from that model can be selected. If the argument is a Flag then only *SENSOR_DEFINEs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any *SENSOR_DEFINEs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of *SENSOR_DEFINEs selected or null if menu cancelled

Example

To select *SENSOR_DEFINEs from model m, flagging those selected with flag f, giving the prompt 'Select *SENSOR_DEFINEs':

```
SensorDefine.Select(f, 'Select *SENSOR_DEFINEs', m);
```

To select *SENSOR_DEFINEs, flagging those selected with flag f but limiting selection to *SENSOR_DEFINEs flagged with flag l, giving the prompt 'Select *SENSOR_DEFINEs':

```
SensorDefine.Select(f, 'Select *SENSOR_DEFINEs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the *SENSOR_DEFINE.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the *SENSOR_DEFINE

Return type

No return value

Example

To set flag f for *SENSOR_DEFINE sd:

```
sd.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of *SENSOR_DEFINEs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing *SENSOR_DEFINEs should be counted. If false or omitted referenced but undefined *SENSOR_DEFINEs will also be included in the total.

Return type

number of *SENSOR_DEFINEs

Example

To get the total number of *SENSOR_DEFINEs in model m:

```
var total = SensorDefine.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the *SENSOR_DEFINEs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all *SENSOR_DEFINEs will be unset in
flag	Flag	Flag to unset on the *SENSOR_DEFINEs

Return type

No return value

Example

To unset the flag f on all the *SENSOR_DEFINEs in model m:

```
SensorDefine.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

SensorDefine object.

Example

To check if SensorDefine property sd.example is a parameter by using the SensorDefine.GetParameter() method:

```
if (sd.ViewParameters().GetParameter(sd.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for *SENSOR_DEFINE. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for *SENSOR_DEFINE sd:

```
sd.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this *SENSOR_DEFINE.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for *SENSOR_DEFINE sd:

```
var xrefs = sd.Xrefs();
```

---

toString()

Description

Creates a string containing the sensor define data in keyword format. Note that this contains the keyword header and the keyword cards. See also SensorDefine.Keyword() and SensorDefine.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for sensor define sd in keyword format

```
var str = sd.toString();
```

SensorSwitch class

The SensorSwitch class gives you access to *SENSOR_SWITCH keyword in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRow(row*[integer]*)
- GetSwitch(row*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRow(row*[integer]*)
- RemoveSwitch(row*[integer]*)
- SetFlag(flag*[Flag]*)
- SetRow(row*[integer]*, data*[Array of data]*)
- SetSwitch(index*[integer]*, data*[object]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

SensorSwitch constants

Name	Description

SensorSwitch.SWITCH	Sensor switch is *SENSOR_SWITCH.
SensorSwitch.SWITCH_CALC_LOGIC	Sensor switch is *SENSOR_SWITCH_CALC-LOGIC.
SensorSwitch.SWITCH_SHELL_TO_VENT	Sensor switch is *SENSOR_SWITCH_SHELL_TO_VENT.

SensorSwitch properties

Name	Type	Description
abid	integer	Airbag ID.
c23	integer/real	Vent coefficient if positive or user defined load curve ID if negative.
exists	logical	true if *SENSOR_SWITCH exists, false if referred to but not defined. (read only)
filtrid	integer	Filter ID.
id	integer	Part set ID or Part ID.
id_flag	logical	Turns _TITLE/_ID ON or OFF. Used only for SensorSwitch.SWITCH_SHELL_TO_VENT.
include	integer	The Include file number that the *SENSOR_SWITCH is in.
itype	integer	0 for Part, 1 for Part Set.
label	integer	SensorSwitch number. The switid property is an alternative name for this.
logic	string	Logic operator.
model	integer	The Model number that the *SENSOR_SWITCH is in.
nrow	integer	Number of Shell Fail Time Cards.
nswit	integer	Number of sensor switch IDs defined (read only). IDs can be positive for "AND", negative ID for "OR". Applicable to SensorSwitch.SWITCH_CALC_LOGIC.
option	constant	SENSOR_SWITCH suffix. Can be SensorSwitch.SWITCH, SensorSwitch.SWITCH_CALC_LOGIC or SensorSwitch.SWITCH_SHELL_TO_VENT.
sensid	integer	ID of the sensor whose value will be compared.
switid	integer	SensorSwitch number. The label property is an alternative name for this.
timwin	real	Trigger status change when the value given by the sensor is less/greater (depending on logic) than value for duration defined by timwin.
title	string	SensorSwitch title. Used only for SensorSwitch.SWITCH_SHELL_TO_VENT.
type	string	This property is deprecated in version R9.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Type used for sensor. Can either be "SENSOR" or "TIME" [deprecated]
value	real	Critical value.

Detailed Description

The SensorSwitch class allows you to create, modify, edit and manipulate *SENSOR_SWITCH. See the documentation below for more details.

Constructor

new SensorSwitch(Option*[constant]*, Model*[Model]*, Switch ID*[integer]*)

Description

Create a new SensorSwitch object.

Arguments

Name	Type	Description
Option	constant	SENSOR_SWITCH suffix. Can be SensorSwitch.SWITCH, SensorSwitch.SWITCH_CALC_LOGIC or SensorSwitch.SWITCH_SHELL_TO_VENT.
Model	Model	Model that *SENSOR_SWITCH will be created in
Switch ID	integer	SensorSwitch id. This is required for the SensorSwitch.SWITCH and SensorSwitch.SWITCH_CALC_LOGIC options and ignored for SensorSwitch.SWITCH_SHELL_TO_VENT.

Return type

SensorSwitch object

Example

To create a new *SENSOR_SWITCH in model m with label 100:

```
var sc = new SensorSwitch(SensorSwitch.SWITCH, m, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a *SENSOR_SWITCH.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the *SENSOR_SWITCH

Return type

No return value

Example

To associate comment c to the *SENSOR_SWITCH ss:

```
ss.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse *SENSOR_SWITCH ss:

```
ss.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the *SENSOR_SWITCH.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the *SENSOR_SWITCH

Return type

No return value

Example

To clear flag f for *SENSOR_SWITCH ss:

```
ss.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the *SENSOR_SWITCH.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

SensorSwitch object

Example

To copy *SENSOR_SWITCH ss into *SENSOR_SWITCH z:

```
var z = ss.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a *SENSOR_SWITCH.

Arguments

Name	Type	Description
Model	Model	Model that the *SENSOR_SWITCH will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

SensorSwitch object (or null if not made)

Example

To start creating a *SENSOR_SWITCH in model m:

```
var ss = SensorSwitch.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a *SENSOR_SWITCH.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the *SENSOR_SWITCH

Return type

No return value

Example

To detach comment c from the *SENSOR_SWITCH ss:

```
ss.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit *SENSOR_SWITCH ss:

```
ss.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for *SENSOR_SWITCH. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give

details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for *SENSOR_SWITCH ss:

```
ss.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first *SENSOR_SWITCH in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first *SENSOR_SWITCH in

Return type

SensorSwitch object (or null if there are no *SENSOR_SWITCHs in the model).

Example

To get the first *SENSOR_SWITCH in model m:

```
var ss = SensorSwitch.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free *SENSOR_SWITCH label in the model. Also see SensorSwitch.LastFreeLabel(), SensorSwitch.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free *SENSOR_SWITCH label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

SensorSwitch label.

Example

To get the first free *SENSOR_SWITCH label in model m:

```
var label = SensorSwitch.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the *SENSOR_SWITCHs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_SWITCHs will be flagged in
flag	Flag	Flag to set on the *SENSOR_SWITCHs

Return type

No return value

Example

To flag all of the *SENSOR_SWITCHs with flag f in model m:

```
SensorSwitch.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the *SENSOR_SWITCH is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the *SENSOR_SWITCH

Return type

true if flagged, false if not.

Example

To check if *SENSOR_SWITCH ss has flag f set on it:

```
if (ss.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each *SENSOR_SWITCH in the model.
Note that ForEach has been designed to make looping over *SENSOR_SWITCHs as fast as possible and so has some limitations.
Firstly, a single temporary SensorSwitch object is created and on each function call it is updated with the current *SENSOR_SWITCH data. This means that you should not try to store the SensorSwitch object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new *SENSOR_SWITCHs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_SWITCHs are in

func	function	Function to call for each *SENSOR_SWITCH
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the *SENSOR_SWITCHs in model m:

```
SensorSwitch.ForEach(m, test);
function test(ss)
{
// ss is SensorSwitch object
}
```

To call function test for all of the *SENSOR_SWITCHs in model m with optional object:

```
var data = { x:0, y:0 };
SensorSwitch.ForEach(m, test, data);
function test(ss, extra)
{
// ss is SensorSwitch object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of SensorSwitch objects for all of the *SENSOR_SWITCHs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_SWITCHs from

Return type

Array of SensorSwitch objects

Example

To make an array of SensorSwitch objects for all of the *SENSOR_SWITCHs in model m

```
var ss = SensorSwitch.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a *SENSOR_SWITCH.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the *SENSOR_SWITCH ss:

```
var comm_array = ss.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of SensorSwitch objects for all of the flagged *SENSOR_SWITCHs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get *SENSOR_SWITCHs from
flag	Flag	Flag set on the *SENSOR_SWITCHs that you want to retrieve

Return type

Array of SensorSwitch objects

Example

To make an array of SensorSwitch objects for all of the *SENSOR_SWITCHs in model m flagged with f

```
var ss = SensorSwitch.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the SensorSwitch object for a *SENSOR_SWITCH ID.

Arguments

Name	Type	Description
Model	Model	Model to find the *SENSOR_SWITCH in
number	integer	number of the *SENSOR_SWITCH you want the SensorSwitch object for

Return type

SensorSwitch object (or null if *SENSOR_SWITCH does not exist).

Example

To get the SensorSwitch object for *SENSOR_SWITCH 100 in model m

```
var ss = SensorSwitch.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a SensorSwitch property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the SensorSwitch.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	*SENSOR_SWITCH property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if SensorSwitch property ss.example is a parameter:

```
Options.property_parameter_names = true;
if (ss.GetParameter(ss.example) ) do_something...
Options.property_parameter_names = false;
```

To check if SensorSwitch property ss.example is a parameter by using the GetParameter method:

```
if (ss.ViewParameters().GetParameter(ss.example) ) do_something...
```

GetRow(row*[integer]*)

Description

Returns the data for a row in the SENSOR_SWITCH_SHELL_TO_VENT.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

An array of numbers containing the row variables SSID, FTIME and C23V.

Example

To get the data for the 2nd row in sensor switch ss:

```
var data = ss.GetRow(1);
```

GetSwitch(row*[integer]*)

Description

Returns switch ID information for *SENSOR_SWITCH_CALC-LOGIC.

Arguments

Name	Type	Description

row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

Object containing sensor switch ID information.

Example

To get the data for the 2nd switch in sensor switch ss:

```
var data = ss.GetSwitch(1);
Message("Switch 2: " + data.swit);
```

Keyword()

Description

Returns the keyword for this *SENSOR_SWITCH. **Note that a carriage return is not added**. See also SensorSwitch.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for SensorSwitch ss:

```
var key = ss.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the *SENSOR_SWITCH. **Note that a carriage return is not added**. See also SensorSwitch.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for sensor switch ss:

```
var cards = ss.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last *SENSOR_SWITCH in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last *SENSOR_SWITCH in

Return type

SensorSwitch object (or null if there are no *SENSOR_SWITCHs in the model).

Example

To get the last *SENSOR_SWITCH in model m:

```
var ss = SensorSwitch.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free *SENSOR_SWITCH label in the model. Also see SensorSwitch.FirstFreeLabel(), SensorSwitch.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free *SENSOR_SWITCH label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

SensorSwitch label.

Example

To get the last free *SENSOR_SWITCH label in model m:

```
var label = SensorSwitch.LastFreeLabel(m);
```

Next()

Description

Returns the next *SENSOR_SWITCH in the model.

Arguments

No arguments

Return type

SensorSwitch object (or null if there are no more *SENSOR_SWITCHs in the model).

Example

To get the *SENSOR_SWITCH in model m after *SENSOR_SWITCH ss:

```
var ss = ss.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) *SENSOR_SWITCH label in the model. Also see SensorSwitch.FirstFreeLabel(), SensorSwitch.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free *SENSOR_SWITCH label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

SensorSwitch label.

Example

To get the next free *SENSOR_SWITCH label in model m:

```
var label = SensorSwitch.NextFreeLabel(m);
```

Previous()

Description

Returns the previous *SENSOR_SWITCH in the model.

Arguments

No arguments

Return type

SensorSwitch object (or null if there are no more *SENSOR_SWITCHs in the model).

Example

To get the *SENSOR_SWITCH in model m before *SENSOR_SWITCH ss:

```
var ss = ss.Previous();
```

RemoveRow(row*[integer]*)

Description

Removes the data for a row in *SENSOR_SWITCH_SHELL_TO_VENT.

Arguments

Name	Type	Description
row	integer	The row you want to remove the data for. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second row of data for sensor switch ss:

```
ss.RemoveRow(1);
```

RemoveSwitch(row*[integer]*)

Description

Removes sensor switch ID from *SENSOR_SWITCH_CALC-LOGIC.

Arguments

Name	Type	Description
row	integer	The sensor switch ID that you want to remove. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second sensor switch ID for sensor switch ss:

```
ss.RemoveSwitch(1);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the *SENSOR_SWITCHs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all *SENSOR_SWITCHs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_SWITCHs in model m, from 1000000:

```
SensorSwitch.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged *SENSOR_SWITCHs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged *SENSOR_SWITCHs will be renumbered in

flag	Flag	Flag set on the *SENSOR_SWITCHs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the *SENSOR_SWITCHs in model m flagged with f, from 1000000:

```
SensorSwitch.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select *SENSOR_SWITCHs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting *SENSOR_SWITCHs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only *SENSOR_SWITCHs from that model can be selected. If the argument is a Flag then only *SENSOR_SWITCHs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any *SENSOR_SWITCHs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of *SENSOR_SWITCHs selected or null if menu cancelled

Example

To select *SENSOR_SWITCHs from model m, flagging those selected with flag f, giving the prompt 'Select *SENSOR_SWITCHs':

```
SensorSwitch.Select(f, 'Select *SENSOR_SWITCHs', m);
```

To select *SENSOR_SWITCHs, flagging those selected with flag f but limiting selection to *SENSOR_SWITCHs flagged with flag l, giving the prompt 'Select *SENSOR_SWITCHs':

```
SensorSwitch.Select(f, 'Select *SENSOR_SWITCHs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the *SENSOR_SWITCH.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the *SENSOR_SWITCH

Return type

No return value

Example

To set flag f for *SENSOR_SWITCH ss:

```
ss.SetFlag(f);
```

SetRow(row*[integer]*, data*[Array of data]*)

Description

Sets the data for a row in *SENSOR_SWITCH_SHELL_TO_VENT.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that row indices start at 0**.
data	Array of data	An array containing the row variables SSID, FTIME and C23V.

Return type

No return value.

Example

To set the second row of data for sensor switch ss to be shell set list 11, time 12.0 and vent coefficient 0.7:

```
var array = [11, 12.0, 0.7];
ss.SetRow(1, array);
```

To append a new row of data (using the same array of values):

```
ss.SetRow(ss.nrow, array);
```

SetSwitch(index*[integer]*, data*[object]*)

Description

Specifies a sensor switch ID for a *SENSOR_SWITCH_CALC-LOGIC.

Arguments

Name	Type	Description
index	integer	The index of the *SENSOR_SWITCH_CALC-LOGIC data to set. **Note that indices start at 0, not 1**. 0 <= index <= nswit
data	object	**Name** / **Type** / **Description** swit / integer / Positive or negative sensor switch id. Object containing sensor swith ID data. Object has the following properties:

Return type

No return value.

Example

To set the value of -10 for sensor switch 5 (indices start with 0) for *SENSOR_SWITCH_CALC-LOGIC s:

```
var data = { swit: -10 };
s.SetSwitch(4, data);
```

To append a new line of data (using the same example values):

```
var data2 = {swit: -10};
s.SetSwitch(b.lines, data2);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of *SENSOR_SWITCHs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing *SENSOR_SWITCHs should be counted. If false or omitted referenced but undefined *SENSOR_SWITCHs will also be included in the total.

Return type

number of *SENSOR_SWITCHs

Example

To get the total number of *SENSOR_SWITCHs in model m:

```
var total = SensorSwitch.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the *SENSOR_SWITCHs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all *SENSOR_SWITCHs will be unset in
flag	Flag	Flag to unset on the *SENSOR_SWITCHs

Return type

No return value

Example

To unset the flag f on all the *SENSOR_SWITCHs in model m:

```
SensorSwitch.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

SensorSwitch object.

Example

To check if SensorSwitch property ss.example is a parameter by using the SensorSwitch.GetParameter() method:

```
if (ss.ViewParameters().GetParameter(ss.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for *SENSOR_SWITCH. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for *SENSOR_SWITCH ss:

```
ss.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this *SENSOR_SWITCH.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for *SENSOR_SWITCH ss:

```
var xrefs = ss.Xrefs();
```

---

toString()

Description

Creates a string containing the sensor switch data in keyword format. Note that this contains the keyword header and the keyword cards. See also SensorSwitch.Keyword() and SensorSwitch.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for sensor switch ss in keyword format

```
var str = ss.toString();
```

Set class

The Set class gives you access to sets in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, type*[constant]*, modal (optional)*[boolean]*)
- First(Model*[Model]*, type*[constant]*)
- FirstFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*)
- GetAll(Model*[Model]*, type*[constant]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*, type*[constant]*)
- GetFromID(Model*[Model]*, set number*[integer]*, type*[constant]*)
- Last(Model*[Model]*, type*[constant]*)
- LastFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*)
- Pick(type*[constant]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*, type (optional)*[constant]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*, type (optional)*[constant]*)
- Select(type*[constant]*, flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*)
- UnsketchAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*)

Member functions

- Add(id1*[integer]*, id2 (only for SEGMENT, _GENERATE, _GENERATE_INCREMENT and _ADD_ADVANCED sets)*[integer]*, id3 (only for SEGMENT and _GENERATE_INCREMENT sets)*[integer]*, id4 (only for SEGMENT sets)*[integer]*)
- AddCollectChild(set*[Set]*)
- AddFlagged(flag*[Flag]*)
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Contains(id*[integer]*)
- Copy(range (optional)*[boolean]*)
- Edit(modal (optional)*[boolean]*)
- Empty()
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetCollectChild(number*[Integer]*)
- GetGeneralData(index*[Integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RebuildCache()
- Remove(id*[integer]*)
- RemoveFlagged(flag*[Flag]*)
- RemoveGeneralData(index*[Integer]*)
- SetFlag(flag*[Flag]*)

- SetGeneralData(index*[Integer]*, data*[Array of data]*)
- Sketch(redraw (optional)*[boolean]*)
- Spool()
- StartSpool(raw (optional)*[boolean]*)
- Unsketch(redraw (optional)*[boolean]*)
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Set constants

Name	Description
Set.2D_SEGMENT	This constant is deprecated in version 11.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Set.SEGMENT_2D instead. [deprecated]
Set.ADD	Set type is *SET_XYZ_ADD.
Set.ALL_TYPES	All set types - used in blanking etc.
Set.BEAM	Set beam type
Set.BOX	Set box type
Set.DISCRETE	Set discrete type
Set.GENERAL	Set type is *SET_XYZ_GENERAL.
Set.GENERATE	Set type is *SET_XYZ_GENERATE.
Set.INTERSECT	Set type is *SET_XYZ_INTERSECT.
Set.MM_GROUP	Set multi-material group type
Set.MODE	Set mode type
Set.NODE	Set node type
Set.PART	Set part type
Set.PART_TREE	Set part tree type
Set.PERI_ LAMINATE	Set Peri Laminate type
Set.SEGMENT	Set segment type
Set.SEGMENT_2D	Set segment 2d type
Set.SHELL	Set shell type
Set.SOLID	Set solid type
Set.TSHELL	Set thick shell type

Set properties

Name	Type	Description
add	logical	If _ADD option is active.
advanced (read only)	logical	If _ADD_ADVANCED option is active.

collect	logical	If _COLLECT option is active. To manage _COLLECT sets PRIMER creates a 'parent' set that can be used to sketch/view all of the items from the _COLLECT sets with the same label. PRIMER then manages each _COLLECT set with the same label as a 'child' of this 'parent' set. Also see collect_children and GetCollectChild. If the collect property is unset for a child collect set then a new label will be assigned for the child set. If the collect property is unset for a parent collect set then all of the child sets will be reassigned new labels.
collect_ children (read only)	integer	The number of child _COLLECT sets if _COLLECT option is active.
colour	Colour	The colour of the set
column (read only)	logical	If _COLUMN option is active.
da1	real	The first default attribute for the set (only valid for Set.NODE, Set.PART, Set.SEGMENT, Set.SEGMENT_2D and Set.SHELL)
da2	real	The second default attribute for the set (only valid for Set.NODE, Set.PART, Set.SEGMENT, Set.SEGMENT_2D and Set.SHELL)
da3	real	The third default attribute for the set (only valid for Set.NODE, Set.PART, Set.SEGMENT, Set.SEGMENT_2D and Set.SHELL)
da4	real	The fourth default attribute for the set (only valid for Set.NODE, Set.PART, Set.SEGMENT, Set.SEGMENT_2D and Set.SHELL)
exists	logical	true if set exists, false if referred to but not defined. (read only)
general	logical	If _GENERAL option is active.
general_lines (read only)	integer	Number of lines of data for _GENERAL set (if _GENERAL option is active).
generate	logical	If _GENERATE option is active.
include	integer	The Include file number that the set is in.
increment (read only)	logical	If _GENERATE_INCREMENT option is active.
intersect	logical	If _INTERSECT option is active.
label	integer	Set number. Also see the sid property which is an alternative name for this.
model	integer	The Model number that the set is in.
sid	integer	Set number. Also see the label property which is an alternative name for this.
smooth (read only)	logical	If _LIST_SMOOTH option is active.
solver	string	Solver to attach to set. Can be "MECH", "CESE", "EM", "ICFD" or blank (only valid for Set.NODE, Set.PART, Set.SEGMENT and Set.SOLID).
title	string	Set title
total (read only)	integer	The total number of items in the set. Note that for _GENERAL and _GENERATE sets this is expensive to compute.
transparency	integer	The transparency of the set (0-100) 0% is opaque, 100% is transparent.
type (read only)	constant	Set type. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Detailed Description

The Set class allows you to create, modify, edit and manipulate sets. See the documentation below for more details.
ECMAScript 6 defines a Set class for Set objects so unfortunately this clashes with the Set class we have defined in PRIMER for the LS-DYNA keyword *SET. By default the Set class is used for the LS-DYNA keyword *SET but this can be changed by using the preference 'set_class' in the preferences editor.
The LS-DYNA keyword *SET class is also available (regardless of whether Set is used for the keyword or ECMAScript 6 Set objects) by using SetK (similarly to Nrb being an alias for the NodalRigidBody class).

Constructor

new Set(Model*[Model]*, sid*[integer]*, type*[constant]*, title (optional)*[string]*, option (optional)*[constant]*)

Description

Create a new Set object.

Arguments

Name	Type	Description
Model	Model	Model that set will be created in
sid	integer	Set number
type	constant	Type of set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL
title (optional)	string	Title for the set
option (optional)	constant	Set type. Can be Set.ADD, Set.INTERSECT, Set.GENERAL or Set.GENERATE

Return type

Set object

Example

To create a new node set in model m with label 100:

```
var s = new Set(m, 100, Set.NODE);
```

To create a new *NODE_SET_ADD in model m with label 101:

```
var s = new Set(m, 101, Set.NODE,"", Set.ADD);
```

Details of functions

Add(id1*[integer]*, id2 (only for SEGMENT, _GENERATE, _GENERATE_INCREMENT and _ADD_ADVANCED sets)*[integer]*, id3 (only for SEGMENT and _GENERATE_INCREMENT sets)*[integer]*, id4 (only for SEGMENT sets)*[integer]*)

Description

Adds an item to the set. **This cannot be used for _COLUMN and _GENERAL sets**. For segment sets four nodes must be given to define a segment to add to the set.

Arguments

Name	Type	Description
id1	integer	id of the item to add to the set (normal, _ADD or _ADD_ADVANCED sets) or Start ID (_GENERATE sets)
id2 (only for SEGMENT, _GENERATE, _GENERATE_INCREMENT and _ADD_ADVANCED sets)	integer	type of the item to add to the set [1-7] (_ADD_ADVANCED sets) or End ID (_GENERATE sets)
id3 (only for SEGMENT and _GENERATE_INCREMENT sets)	integer	Increment for _GENERATE_INCREMENT sets, otherwise id of the item to add to the set.
id4 (only for SEGMENT sets)	integer	id of the item to add to the set.

Return type

No return value

Example

To add node 10 to node set ns:

```
ns.Add(10);
```

To add segment 10, 11, 12, 13 to segment set ss:

```
ss.Add(10, 11, 12, 13);
```

AddCollectChild(set[*Set*])

Description

Adds a child collect set to the set. The child set label will be changed to be the same as the parent set and it will become a child. Also see Set.collect_children and Set.GetCollectChild.

Arguments

Name	Type	Description
set	Set	Set to be added as a child collect set.

Return type

No return value

Example

To make set ns2 to node set ns:

```
ns.AddCollectChild(ns2);
```

AddFlagged(flag[*Flag*])

Description

Adds flagged items to the set. **This cannot be used for _GENERAL or _GENERATE sets** and **cannot be used for segment sets**

Arguments

Name	Type	Description
flag	Flag	Flag for items to add to the set

Return type

No return value

Example

To add any nodes flagged with flag f to node set ns:

```
ns.AddFlagged(f);
```

BlankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be blanked in
type (optional)	constant	Type of sets to blank. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. Set.ALL_TYPES. If omitted sets of all types will be blanked.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sets in model m:

```
Set.BlankAll(m);
```

To blank all of the node sets in model m:

```
Set.BlankAll(m, Set.NODE);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sets will be blanked in
flag	Flag	Flag set on the sets that you want to blank
type (optional)	constant	Type of sets to blank. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. Set.ALL_TYPES. If set, only flagged sets of this type will be blanked. If omitted flagged sets of all types will be blanked.

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the sets in model m flagged with f:

```
Set.BlankFlagged(m, f);
```

To blank all of the node sets in model m flagged with f:

```
Set.BlankFlagged(m, f, Set.NODE);
```

Blanked()

Description

Checks if the set is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if set s is blanked:

```
if (s.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

No return value

Example

To browse set s:

```
var s.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the set.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the set

Return type

No return value

Example

To clear flag f for set s:

```
s.ClearFlag(f);
```

Contains(id*[integer]*)

Description

Checks if an item is in the set. **This cannot be used for ADD_ADVANCED, _GENERAL or _GENERATE sets** and **cannot be used for segment sets**

Arguments

Name	Type	Description
id	integer	id of the item to check.

Return type

true if item is in set, false if not

Example

To see if node 10 is in node set ns:

```
if (ns.Contains(10) )
{
    do something...
}
```

Copy(range (optional)*[boolean]*)

Description

Copies the set.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. To set current include, use _Include.MakeCurrentLayer() .

Return type

Set object

Example

To copy node net ns into node net ns1:

```
var ns1 = ns.Copy();
```

Create(Model*[Model]*, type*[constant]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a set.

Arguments

Name	Type	Description
Model	Model	Model that the set will be created in
type	constant	Type of the set that you want to create. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Set object (or null if not made)

Example

To start creating a node set in model m:

```
var s = Set.Create(m, Set.NODE);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel to edit the set.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

No return value

Example

To edit set s:

```
var s.Edit();
```

Empty()

Description

Removes all items from the set. **This cannot be used for _GENERATE sets** and **cannot be used for segment sets**

Arguments

No arguments

Return type

No return value

Example

To remove all nodes from node set ns:

```
ns.Empty(f);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for a set. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for set s:

```
s.Error("My custom error");
```

First(Model*[Model]*, type*[constant]*) [static]

Description

Returns the first set in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first set in
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

Set object (or null if there are no sets in the model).

Example

To get the first node set in model m:

```
var n = Set.First(m, Set.NODE);
```

FirstFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free set label in the model. Also see Set.LastFreeLabel(), Set.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free Set label in
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Set label.

Example

To get the first free node set label in model m:

```
var label = Set.FirstFreeLabel(m, Set.NODE);
```

FlagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*) [static]

Description

Flags all of the sets in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be flagged in
flag	Flag	Flag to set on the sets
type (optional)	constant	Type of the set. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. If set, only sets of this type will be flagged. If omitted sets of all types will be flagged.

Return type

No return value

Example

To flag all of the node sets with flag f in model m:

```
Set.FlagAll(m, f, Set.NODE);
```

Flagged(flag*[Flag]*)

Description

Checks if the set is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the set

Return type

true if flagged, false if not.

Example

To check if set s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

GetAll(Model*[Model]*, type*[constant]*) [static]

Description

Returns an array of Set objects for all of the sets in a models in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sets from
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

Array of Set objects

Example

To make an array of Set objects for all of the node sets in model m

```
var n = Set.GetAll(m, Set.NODE);
```

GetCollectChild(number*[Integer]*)

Description

Returns a child collect set. Also see Set.collect_children and Set.AddCollectChild.

Arguments

Name	Type	Description
number	Integer	The index of the child collect set to return. **Note that indices start at 0, not 1**

Return type

Set object

Example

To loop over the child collect sets for set ns:

```
if (ns.collect)
{
    for (i=0; i<ns.collect_children; i++)
        var child = ns.GetCollectChild(i);
}
```

GetFlagged(Model*[Model]*, flag*[Flag]*, type*[constant]*) [static]

Description

Returns an array of Set objects for all of the flagged sets in a models in Primer

Arguments

Name	Type	Description
Model	Model	Model to get sets from
flag	Flag	Flag set on the set that you want to retrieve
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

Array of Set objects

Example

To make an array of Set objects for all of the node sets in model m flagged with f

```
var n = Set.GetFlagged(m, f, Set.NODE);
```

GetFromID(Model*[Model]*, set number*[integer]*, type*[constant]*) [static]

Description

Returns the Set object for a set ID.

Arguments

Name	Type	Description
Model	Model	Model to find the set in
set number	integer	number of the set you want the Set object for
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

Set object (or null if set does not exist).

Example

To get the Set object for node set 100 in model m

```
var n = Set.GetFromID(m, 100, Set.NODE);
```

GetGeneralData(index*[Integer]*)

Description

Returns a line of data for a GENERAL set.

Arguments

Name	Type	Description
index	Integer	The index of the GENERAL data to return. **Note that indices start at 0, not 1**. 0 <= index < general_lines

Return type

Array containing data.

Example

To loop over the lines of general data sets for set s:

```
if (s.general)
{
    for (i=0; i<s.general_lines; i++)
        var data = s.GetGeneralData(i);
}
```

Keyword()

Description

Returns the keyword for this set (*SET_NODE etc). **Note that a carriage return is not added**. See also Set.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for set s:

```
var key = s.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the set. **Note that a carriage return is not added**. See also Set.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for set s:

```
var cards = s.KeywordCards();
```

Last(Model*[Model]*, type*[constant]*) [static]

Description

Returns the last set in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last set in
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

Set object (or null if there are no sets in the model).

Example

To get the last node set in model m:

```
var n = Set.Last(m, Set.NODE);
```

LastFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free set label in the model. Also see Set.FirstFreeLabel(), Set.NextFreeLabel() and Model.LastFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get last free Set label in
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Set label.

Example

To get the last free node set label in model m:

```
var label = Set.LastFreeLabel(m, Set.NODE);
```

Next()

Description

Returns the next set in the model.

Arguments

No arguments

Return type

Set object (or null if there are no more sets in the model).

Example

To get the set in model m after set n:

```
var n = n.Next();
```

NextFreeLabel(Model*[Model]*, type*[constant]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free set label in the model. Also see Set.FirstFreeLabel(), Set.LastFreeLabel() and Model.NextFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get next free Set label in
type	constant	Type of the set. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Set label.

Example

To get the next free node set label in model m:

```
var label = Set.NextFreeLabel(m, Set.NODE);
```

Pick(type*[constant]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a set.

Arguments

Name	Type	Description
type	constant	Type of sets to pick. Can be Set.BEAM, Set.BOX, Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL.
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sets from that model can be picked. If the argument is a Flag then only sets that are flagged with *limit* can be selected. If omitted, or null, any sets from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.

button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Set object (or null if not picked)

Example

To pick a node set from model m giving the prompt 'Pick set from screen':

```
var n = Set.Pick(Set.NODE, 'Pick set from screen', m);
```

Previous()

Description

Returns the previous set in the model.

Arguments

No arguments

Return type

Set object (or null if there are no more sets in the model).

Example

To get the set in model m before this one:

```
var s = s.Previous();
```

RebuildCache()

Description

Rebuilds the cache for a set. As sets can be built using complex combinations of _GENERAL, _ADD, _INTERSECT options etc Primer creates a 'cache' for the set to speed up set drawing and usage. During normal interactive use this cache is rebuilt as necessary but in JavaScript it is possible for the cache to become out of date (e.g. you change a box position in JavaScript that is used by a *SET_GENERAL). Calling this forces the cache to be rebuilt.

Arguments

No arguments

Return type

No return type

Example

To rebuild the cache for set s:

```
s.RebuildCache();
```

Remove(id*[integer]*)

Description

Removes an item from the set. If the item is not in the set nothing is done. **This cannot be used for ADD_ ADVANCED, _COLUMN, _GENERAL or _GENERATE sets** and **cannot be used for segment sets**

Arguments

Name	Type	Description
id	integer	id of the item to remove from the set.

Return type

No return value

Example

To remove node 10 from node set ns:

```
ns.Remove(10);
```

RemoveFlagged(flag*[Flag]*)

Description

Removes flagged items from the set. **This cannot be used for _GENERAL or _GENERATE sets** and **cannot be used for segment sets**

Arguments

Name	Type	Description
flag	Flag	Flag for items to remove from the set

Return type

No return value

Example

To remove any nodes flagged with flag f from node set ns:

```
ns.RemoveFlagged(f);
```

RemoveGeneralData(index*[Integer]*)

Description

Removes a line of data from a GENERAL set.

Arguments

Name	Type	Description
index	Integer	The index of the GENERAL data to remove. **Note that indices start at 0, not 1**. 0 <= index < general_lines

Return type

No return value

Example

To remove the first line of general data sets for set s:

```
if (s.general)
{
    s.RemoveGeneralData(0);
}
```

RenumberAll(Model*[Model]*, start*[integer]*, type (optional)*[constant]*) [static]

Description

Renumbers all of the sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be renumbered in
start	integer	Start point for renumbering
type (optional)	constant	Type of sets to renumber. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. If omitted sets of all types will be blanked.

Return type

No return value

Example

To renumber all of the sets in model m, from 1000000:

```
Set.RenumberAll(m, 1000000);
```

To renumber all of the node sets in model m, from 1000000:

```
Set.RenumberAll(m, 1000000, Set.NODE);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*, type (optional)*[constant]*) [static]

Description

Renumbers all of the flagged sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sets will be renumbered in
flag	Flag	Flag set on the sets that you want to renumber
start	integer	Start point for renumbering
type (optional)	constant	Type of sets to renumber. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. If omitted sets of all types will be blanked.

Return type

No return value

Example

To renumber all of the sets in model m flagged with f, from 1000000:

```
Set.RenumberFlagged(m, f, 1000000);
```

To renumber all of the node sets in model m flagged with f, from 1000000:

```
Set.RenumberFlagged(m, f, 1000000, Set.NODE);
```

Select(type*[constant]*, flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select sets using standard PRIMER object menus.

Arguments

Name	Type	Description
type	constant	Type of sets to pick. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL.
flag	Flag	Flag to use when selecting sets
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only sets from that model can be selected. If the argument is a Flag then only sets that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any sets from any model can be selected.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of items selected or null if menu cancelled

Example

To select node sets from model m, flagging those selected with flag f, giving the prompt 'Select sets':

```
Set.Select(Set.NODE, f, 'Select sets', m);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the set.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the set

Return type

No return value

Example

To set flag f for set s:

```
s.SetFlag(f);
```

SetGeneralData(index*[Integer]*, data*[Array of data]*)

Description

Sets a line of data for a GENERAL set.

Arguments

Name	Type	Description
index	Integer	The index of the GENERAL data to set. **Note that indices start at 0, not 1**. 0 <= index <= general_lines
data	Array of data	Array containing GENERAL data to set.

Return type

No return value.

Example

To add nodes inside boxes 1, 2 and 3 as a new line of data to node general set s:

```
var data = [ "BOX", 1, 2, 3];
s.SetGeneralData(s.general_lines, data);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the set. The set will be sketched until you either call Set.Unsketch(), Set.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the set is sketched. If omitted redraw is true. If you want to sketch several sets and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch set s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged sets will be sketched in
flag	Flag	Flag set on the sets that you want to sketch
type (optional)	constant	Type of sets to sketch. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. Set.ALL_TYPES. If set, only flagged sets of this type will be sketched. If omitted flagged sets of all types will be sketched.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is true. If you want to do several (un)sketches and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To sketch all of the sets in model m flagged with f:

```
Set.SketchFlagged(m, f);
```

To sketch all of the node sets in model m flagged with f:

```
Set.SketchFlagged(m, f, Set.NODE);
```

Spool()

Description

Spools a set, entry by entry. See also Set.StartSpool

Arguments

No arguments

Return type

For Set.SEGMENT returns an array containing node IDs, for all other set types returns the ID of the item. Returns 0 if no more items

Example

To spool set s:

```
var id;
s.StartSpool();
while ( (id = s.Spool()) )
{
    do something...
}
```

Note that the extra brackets around the assignment in the example are only required to prevent a warning when compiling the script using strict mode in the debugger. This check is to help the user find cases where they accidentally typed = but actually meant ==. Adding the extra brackets stops the check from being done.

StartSpool(raw (optional)*[boolean]*)

Description

Starts a set spooling operation. See also Set.Spool

Arguments

Name	Type	Description
raw (optional)	boolean	If true then the raw data from _GENERATE, _ADD and _INTERSECT sets will be returned instead of expanding the data ranges or child set contents. If omitted raw will be false.

Return type

No return value

Example

To start spooling set s:

```
s.StartSpool();
```

UnblankAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be unblanked in
type (optional)	constant	Type of sets to unblank. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. Set.ALL_TYPES. If omitted sets of all types will be blanked.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sets in model m:

```
Set.UnblankAll(m);
```

To unblank all of the node sets in model m:

```
Set.UnblankAll(m, Set.NODE);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged sets will be unblanked in
flag	Flag	Flag set on the sets that you want to unblank
type (optional)	constant	Type of sets to unblank. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. Set.ALL_TYPES. If set, only flagged sets of this type will be unblanked. If omitted flagged sets of all types will be unblanked.
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the sets in model m flagged with f:

```
Set.UnblankFlagged(m, f);
```

To unblank all of the node sets in model m flagged with f:

```
Set.UnblankFlagged(m, f, Set.NODE);
```

UnflagAll(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*) [static]

Description

Unsets a defined flag on all of the sets in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all sets will be unset in
flag	Flag	Flag to unset on the sets
type (optional)	constant	Type of the set. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.MODE, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL

Return type

No return value

Example

To unset the flag f on all the sets in model m:

```
Set.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the set.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the set is unsketched. If omitted redraw is true. If you want to unsketch several sets and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch set s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all sets.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be unsketched in
type (optional)	constant	Type of sets to unsketch. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. If omitted sets of all types will be unsketched.
redraw (optional)	boolean	If model should be redrawn or not after the sets are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sets in model m:

```
Set.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, type (optional)*[constant]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged sets.

Arguments

Name	Type	Description
Model	Model	Model that all sets will be unsketched in
flag	Flag	Flag set on the sets that you want to unsketch
type (optional)	constant	Type of sets to unsketch. Can be Set.BEAM, Set.BOX Set.DISCRETE, Set.MM_GROUP, Set.NODE, Set.PART, Set.PART_TREE, Set.PERI_LAMINATE, Set.SEGMENT, Set.SEGMENT_2D, Set.SHELL, Set.SOLID or Set.TSHELL. If omitted sets of all types will be unsketched.
redraw (optional)	boolean	If model should be redrawn or not after the sets are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all sets in model m flagged with f:

```
Set.UnsketchFlagged(m, f);
```

To unsketch all of the node sets in model m flagged with f:

```
Set.UnsketchFlagged(m, f, Set.NODE);
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for a set. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for set s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this set.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for this set:

```
var xrefs = s.Xrefs();
```

---

toString()

Description

Creates a string containing the set data in keyword format. Note that this contains the keyword header and the keyword cards. See also Set.Keyword() and Set.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for set n in keyword format

```
var s = n.toString();
```

Termination class

The Termination class gives you access to *TERMINATION_XXXX cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Termination constants

Constants for Direction for Degrees of Freedom (field: dof)

Name	Description
Termination.DOF_X	Degree of freedom in X direction for Force magnitude. Used for Termination.CONTACT.
Termination.DOF_Y	Degree of freedom in Y direction for Force magnitude. Used for Termination.CONTACT.
Termination.DOF_Z	Degree of freedom in Z direction for Force magnitude. Used for Termination.CONTACT.

Constants for Global Direction for Stop Criterion (field: stop)

Name	Description
Termination.STOP_MAG	Stop if displacement magnitude is exceeded. Used for Termination.BODY or Termination.NODE.
Termination.STOP_X	Stop criterion is in Global X direction. Used for Termination.BODY or Termination.NODE.
Termination.STOP_Y	Stop criterion is in Global Y direction. Used for Termination.BODY or Termination.NODE.
Termination.STOP_Z	Stop criterion is in Global X direction. Used for Termination.BODY or Termination.NODE.

Constants for Type of Keyword

Name	Description
Termination.BODY	TERMINATION is *TERMINATION_BODY.
Termination.CONTACT	TERMINATION is *TERMINATION_CONTACT.
Termination.CURVE	TERMINATION is *TERMINATION_CURVE.
Termination.DELETED_SHELLS	TERMINATION is *TERMINATION_DELETED_SHELLS.
Termination.DELETED_SHELLS_SET	TERMINATION is *TERMINATION_DELETED_SHELLS_SET.
Termination.DELETED_SOLIDS	TERMINATION is *TERMINATION_DELETED_SOLIDS.
Termination.DELETED_SOLIDS_SET	TERMINATION is *TERMINATION_DELETED_SOLIDS_SET.
Termination.NODE	TERMINATION is *TERMINATION_NODE.
Termination.SENSOR	TERMINATION is *TERMINATION_SENSOR.

Termination properties

Name	Type	Description
actTime	real	Activation time value. Used for Termination.CONTACT or Termination.CURVE.
dof	integer	Directions to consider for Force Magnitude. Valid values are: Termination.DOF_X or Termination.DOF_Y or Termination.DOF_Z. Used for Termination.CONTACT.
duration	real	Time duration of null resultant force prior to termination. Used for Termination.CONTACT.
exists	logical	true if termination exists, false if referred to but not defined. (read only)
id	integer	Can be Part or NRBC ID based on ptype value for Termination.BODY, OR Contact ID for Termination.CONTACT, OR Node ID for Termination.NODE, OR Curve ID for Termination.CURVE, OR Part for Termination.DELETED_SHELLS or Termination.DELETED_SOLIDS, OR Part Set ID for Termination.DELETED_SHELLS_SET or Termination.DELETED_SOLIDS_SET, OR Sensor Switch ID for Termination.SENSOR.
include	integer	The Include file number that the termination is in.
maxc	real	Maximum (most positive) displacement. If value is 0.0, it is set to 1.0e21. Value should be more than minc. Used for Termination.BODY or Termination.NODE.
minc	real	Minimum (most negative) displacement. If value is 0.0, it is set to -1.0e21. Value should be less than maxc. Used for Termination.BODY or Termination.NODE.
model	integer	The Model number that the termination is in.
numDeletedElems	integer	Number of elements that must be deleted for the specified Part ID's, before an error termination occurs. Used for Termination.DELETED_SHELLS_SET or Termination.DELETED_SOLIDS_SET.

ptype	integer	Gives the type of Part for Termination.BODY. Values can be 0 for Part or 1 for NRBC (read only)
stop	integer	Stop Criterion. Valid values are: Termination.STOP_X or Termination.STOP_Y or Termination.STOP_Z or Termination.STOP_MAG. Used for Termination.BODY or Termination.NODE.
threshold	real	Any measured force magnitude below or equal to this specified threshold is taken as a null force. Used for Termination.CONTACT.
type	integer	Gives the type of *TERMINATION keyword (read only).

Detailed Description

The Termination class allows you to create, modify, edit and manipulate termination cards. See the documentation below for more details.

Constructor

new Termination(Model*[Model]*, Type*[constant]*, id *[integer]*)

Description

Create a new Termination object.

Arguments

Name	Type	Description
Model	Model	Model that termination will be created in
Type	constant	Specify the type of Termination (Can be Termination.BODY or Termination.CONTACT or Termination.CURVE or Termination.DELETED_SHELLS or Termination.DELETED_SOLIDS or Termination.NODE or Termination.SENSOR).
id	integer	Can be Part ID for Termination.BODY or Termination.DELETED_SHELLS or Termination.DELETED_SOLIDS, OR Contact ID for Termination.CONTACT, OR Node ID for Termination.NODE, OR Curve ID for Termination.CURVE, OR Part Set ID for Termination.DELETED_SHELLS_SET or Termination.DELETED_SOLIDS_SET, OR Sensor Switch ID for Termination.SENSOR.

Return type

Termination object

Example

To create a new termination in model m, type BODY, part id 100:

```
var term = new Termination(m, Termination.BODY, 100);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a termination.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the termination

Return type

No return value

Example

To associate comment c to the termination term:

```
term.AssociateComment(c);
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse termination term:

```
term.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the termination.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the termination

Return type

No return value

Example

To clear flag f for termination term:

```
term.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the termination.

Arguments

Name	Type	Description

range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Termination object

Example

To copy termination term into termination z:

```
var z = term.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create an Termination definition.

Arguments

Name	Type	Description
Model	Model	Model that the Termination will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Termination object (or null if not made)

Example

To start creating an termination in model m:

```
var term = Termination.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a termination.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the termination

Return type

No return value

Example

To detach comment c from the termination term:

```
term.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit termination term:

```
term.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for termination. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for termination term:

```
term.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first termination in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first termination in

Return type

Termination object (or null if there are no terminations in the model).

Example

To get the first termination in model m:

```
var term = Termination.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the terminations in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all terminations will be flagged in
flag	Flag	Flag to set on the terminations

Return type

No return value

Example

To flag all of the terminations with flag f in model m:

```
Termination.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the termination is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the termination

Return type

true if flagged, false if not.

Example

To check if termination term has flag f set on it:

```
if (term.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each termination in the model.
Note that ForEach has been designed to make looping over terminations as fast as possible and so has some limitations.
Firstly, a single temporary Termination object is created and on each function call it is updated with the current termination data. This means that you should not try to store the Termination object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new terminations inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all terminations are in
func	function	Function to call for each termination

extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the terminations in model m:

```
Termination.ForEach(m, test);
function test(term)
{
// term is Termination object
}
```

To call function test for all of the terminations in model m with optional object:

```
var data = { x:0, y:0 };
Termination.ForEach(m, test, data);
function test(term, extra)
{
// term is Termination object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Termination objects for all of the terminations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get terminations from

Return type

Array of Termination objects

Example

To make an array of Termination objects for all of the terminations in model m

```
var term = Termination.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a termination.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the termination term:

```
var comm_array = term.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Termination objects for all of the flagged terminations in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get terminations from
flag	Flag	Flag set on the terminations that you want to retrieve

Return type

Array of Termination objects

Example

To make an array of Termination objects for all of the terminations in model m flagged with f

```
var term = Termination.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Termination object for a termination ID.

Arguments

Name	Type	Description
Model	Model	Model to find the termination in
number	integer	number of the termination you want the Termination object for

Return type

Termination object (or null if termination does not exist).

Example

To get the Termination object for termination 100 in model m

```
var term = Termination.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Termination property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Termination.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	termination property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Termination property term.example is a parameter:

```
Options.property_parameter_names = true;
if (term.GetParameter(term.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Termination property term.example is a parameter by using the GetParameter method:

```
if (term.ViewParameters().GetParameter(term.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this Termination (*TERMINATION_xxxx) **Note that a carriage return is not added**. See also Termination.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for Termination termination:

```
var key = Termination.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Termination. **Note that a carriage return is not added**. See also Termination.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Termination termination:

```
var cards = Termination.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last termination in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last termination in

Return type

Termination object (or null if there are no terminations in the model).

Example

To get the last termination in model m:

```
var term = Termination.Last(m);
```

Next()

Description

Returns the next termination in the model.

Arguments

No arguments

Return type

Termination object (or null if there are no more terminations in the model).

Example

To get the termination in model m after termination term:

```
var term = term.Next();
```

Previous()

Description

Returns the previous termination in the model.

Arguments

No arguments

Return type

Termination object (or null if there are no more terminations in the model).

Example

To get the termination in model m before termination term:

```
var term = term.Previous();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select terminations using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting terminations
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only terminations from that model can be selected. If the argument is a Flag then only terminations that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any terminations can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of terminations selected or null if menu cancelled

Example

To select terminations from model m, flagging those selected with flag f, giving the prompt 'Select terminations':

```
Termination.Select(f, 'Select terminations', m);
```

To select terminations, flagging those selected with flag f but limiting selection to terminations flagged with flag l, giving the prompt 'Select terminations':

```
Termination.Select(f, 'Select terminations', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the termination.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the termination

Return type

No return value

Example

To set flag f for termination term:

```
term.SetFlag(f);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of terminations in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing terminations should be counted. If false or omitted referenced but undefined terminations will also be included in the total.

Return type

number of terminations

Example

To get the total number of terminations in model m:

```
var total = Termination.Total(m);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the terminations in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all terminations will be unset in
flag	Flag	Flag to unset on the terminations

Return type

No return value

Example

To unset the flag f on all the terminations in model m:

```
Termination.UnflagAll(m, f);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Termination object.

Example

To check if Termination property term.example is a parameter by using the Termination.GetParameter() method:

```
if (term.ViewParameters().GetParameter(term.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for termination. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for termination term:

```
term.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this termination.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for termination term:

```
var xrefs = term.Xrefs();
```

---

toString()

Description

Creates a string containing the Termination data in keyword format. Note that this contains the keyword header and the keyword cards. See also Termination.Keyword() and Termination.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Termination termination in keyword format

```
var term = termination.toString();
```

Attached class

The Attached class contains constants and static functions relating to the Attached() member function from the Model class. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Beam3rdNodes(Setting*[boolean]*)
- BeamPid(Setting*[boolean]*)
- Deformable(Setting*[constant]*)
- FlagPart(Setting*[boolean]*)
- Recursive(Setting*[boolean]*, Number (optional)*[integer]*)
- Rigid(Setting*[constant]*)
- SetEntity(Type*[string]*, Setting*[boolean]*)
- TiedContacts(Setting*[boolean]*)

Attached constants

Name	Description
Attached.SINGLE	Find attached option - find attached through single elements only
Attached.WHOLE	Find attached option - find through whole attached part

Detailed Description

The Attached class staic functions are used to set options for the find attached feature in Primer. Once set, these settings are used when using the Attached() member function from the Model class

Details of functions

Beam3rdNodes(Setting*[boolean]*) [static]

Description

Sets the find attached option for beam 3rd nodes on or off

Arguments

Name	Type	Description
Setting	boolean	If true beam 3rd nodes are considered for find attached, if false, they are not.

Return type

No return value

Example

To set the 3rd node option to on:

```
Attached.Beam3rdNodes(true);
```

BeamPid(Setting*[boolean]*) [static]

Description

Sets the find attached option for beam pid on or off

Arguments

Name	Type	Description
Setting	boolean	If true beam pid's are considered for find attached, if false, they are not.

Return type

No return value

Example

To set the beam pid option to on:

```
Attached.BeamPid(true);
```

Deformable(Setting*[constant]*) [static]

Description

Sets the deformable option for find attached

Arguments

Name	Type	Description
Setting	constant	Option. Can be Attached.WHOLE, Attached.SINGLE

Return type

No return value

Example

To set the deformable option to find attached through the whole part:

```
Attached.Deformable(Attached.WHOLE);
```

FlagPart(Setting*[boolean]*) [static]

Description

Sets an option to flag parts after a find attached if any elements within that part are flagged

Arguments

Name	Type	Description
Setting	boolean	If true, parts are flagged after a find attached if any elements within that part are flagged, if false, they are not.

Return type

No return value

Example

To set the flag part option to on:

```
Attached.FlagPart(true);
```

Recursive(Setting*[boolean]*, Number (optional)*[integer]*) [static]

Description

Sets the find attached option for recursive on or off

Arguments

Name	Type	Description
Setting	boolean	If true recursive is on, if false, it is off.
Number (optional)	integer	Option to set the number of find attached iterations used when the recursive option is set.

Return type

No return value

Example

To set the recursive option to on:

```
Attached.Recursive(true);
```

Rigid(Setting*[constant]*) [static]

Description

Sets the rigid option for find attached

Arguments

Name	Type	Description
Setting	constant	Option. Can be Attached.WHOLE, Attached.SINGLE

Return type

No return value

Example

To set the rigid option to find attached through the whole part:

```
Attached.Rigid(Attached.WHOLE);
```

SetEntity(Type*[string]*, Setting*[boolean]*) [static]

Description

Sets entity to be on or off to find attached through.

Arguments

Name	Type	Description
Type	string	The type of the item to switch on or off (for a list of types see Appendix I of the PRIMER manual).

Setting	boolean	If true you turn the entity switch on, if false you turn it off.

Return type

No return value

Example

To set the SHELL switch to on so that when you run a find attached you fnd attached through shells:

```
Attached.SetEntity("SHELL", true);
```

TiedContacts(Setting*[boolean]*) [static]

Description

Sets the find attached option for tied contacts on or off

Arguments

Name	Type	Description
Setting	boolean	If true tied contacts are considered for find attached, if false, they are not.

Return type

No return value

Example

To set the tied contacts option to on:

```
Attached.TiedContacts(true);
```

Belt class

The Belt class gives you access to belt fitting in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SetMeshingLabels(entity_type*[constant]*, label_value*[integer]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Fit()
- Flagged(flag*[Flag]*)
- Generate()
- GetComments()
- GetMesh(index*[integer]*)
- GetParameter(prop*[string]*)
- GetPoint(index*[integer]*)
- InsertPoint(index*[integer]*, position*[integer]*, data*[object]*)
- Next()
- Previous()
- RemovePoint(index*[integer]*)
- SetFlag(flag*[Flag]*)
- SetMesh(index*[integer]*, data*[object]*)
- SetPoint(index*[integer]*, data*[object]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)

- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

Belt constants

Constants for Mesh segments

Name	Description
Belt.MSEG_B1_ONLY	Old style all 1D belt
Belt.MSEG_B2_ONLY	Old style all 2D belt
Belt.MSEG_BD_NEW	Indicates new mode. **This must be set before any of the new style constants can be used**
Belt.MSEG_CE_1D	New style 1D at centre
Belt.MSEG_CE_2D	New style 2D at centre
Belt.MSEG_CE_SH	New style SH at centre
Belt.MSEG_E1_1D	New style 1D at end 1
Belt.MSEG_E1_2D	New style 2D at end 1
Belt.MSEG_E1_SH	New style shells at end 1
Belt.MSEG_E2_1D	New style 1D at end 2
Belt.MSEG_E2_2D	New style 2D at end 2
Belt.MSEG_E2_SH	New style shells at end 2
Belt.MSEG_MIX_SB1	Old style 1D at ends, shells in middle
Belt.MSEG_MIX_SB2	Old style 2D at ends, shells in middle
Belt.MSEG_SH_ONLY	Old style all shell belt

Constants for Meshing start Labels

Name	Description
Belt.MESH_2D_SLIPRING_SET_NODE	Set meshing start Labels for 2D slipring node sets
Belt.MESH_ALL	Set meshing start Labels for everything used in the seatbelt definition
Belt.MESH_NODE	Set meshing start Labels for nodes
Belt.MESH_NRBC	Set meshing start Labels for nodal rigid bodies
Belt.MESH_RETRACTOR	Set meshing start Labels for retractors
Belt.MESH_SEATBELT	Set meshing start Labels for seatbelt elements
Belt.MESH_SET_NODE	Set meshing start Labels for node sets
Belt.MESH_SET_PART	Set meshing start Labels for part sets
Belt.MESH_SET_SHELL	Set meshing start Labels for shell sets
Belt.MESH_SHELL	Set meshing start Labels for shells
Belt.MESH_SLIPRING	Set meshing start Labels for slipring elements
Belt.MESH_XSEC	Set meshing start Labels for Database cross sections

Constants for Path point fixity

Name	Description
Belt.B_POST_ SLIPRING	There is a B-Post slipring at this point.
Belt.FIXED	Point is fixed
Belt.FREE_SLIPRING	There is a free (eg pelvis) slipring at this point.
Belt.KNOWN	The belt path is known to pass through this point
Belt.RETRACTOR	There is a retractor at this point
Belt.SLIPRING	There is a slipring at this point. (Deprecated from V12 onwards, use FREE_SLIPRING or B_POST_SLIPRING instead)
Belt.TWIST	Point has twist vectors or twist nodes defined
Belt.XSEC	There is a database cross section at this point

Constants for Path point insertion

Name	Description
Belt.INSERT_AFTER	Insert after given path point.
Belt.INSERT_BEFORE	Insert before given path point.

Belt properties

Name	Type	Description
acuteAngle	real	Limiting angle to be considered "acute" (0 means 90)
curvature	real	Maximum permitted transverse belt curvature in degrees
elemSet	integer	Set of shell or 2D seatbelt elements. Only created if the option to generate a contact for the belt is used (read only)
exists	logical	true if belt exists, false if referred to but not defined. (read only)
friction	real	Transverse friction coefficient
id	integer	Belt number. Also see the label property which is an alternative name for this.
include	integer	The Include file number that the belt is in.
iterations	integer	The number of fitting iterations between contact bucket resorts
label	integer	Belt number. Also see the id property which is an alternative name for this.
length	real	The characteristic length of each belt element
meshSegs	integer	Number of mesh segments defined (read only)
model	integer	The Model number that the belt is in.
n2sContact	integer	Nodes to Surface contact used between nodes on 1D belt elements and dummy structure. Only used if the option to create a contact between belt and dummy "structure" has been used (read only)
nodeSet	integer	Set of all nodes in seatbelt. Only created if the option to generate a contact for the belt is used (read only)
nrbFirst	integer	First nodal rigid body ID (read only)
nrbLast	integer	Last nodal rigid body ID (read only)
nsboSet	integer	Set of nodes on 1D seatbelt elements only. Only created if the option to generate a contact for the belt is used (read only)
overlap	real	Fraction by which facets are extended during contact checking to stop nodes "falling into gaps"

parts	integer	Part set ID defining structure. Note that if you are creating the seatbelt definition from scratch in JavaScript you **must** define a shell, solid or thick shell set.
penetration	real	Maximum penetration distance considered for contact into solid and thick shell elements
pidShell	integer	The part ID for any 2D seatbelt elements
pid_1d	integer	The part ID for any 1D seatbelt elements
pid_2d	integer	The part ID for any 2D seatbelt elements
points	integer	Number of path points defined (read only)
projection	real	Initial projection distance by which belt path is "thrown outwards" at start of fitting
psiShell	real	Optional orthotropic angle for any shell elements
psi_2d	real	Optional orthotropic angle for any 2D seatbelt elements
retractorFirst	integer	First retractor ID (read only)
retractorLast	integer	Last retractor ID (read only)
rows	integer	The number of rows of 2D elements across the belt
s2sContact	integer	Surface to Surface contact used between shell/2D belt elements and dummy structure. Only used if the option to create a contact between belt and dummy "structure" has been used (read only)
seatbeltFirst	integer	First 1D seatbelt ID (read only)
seatbeltLast	integer	Last 1D seatbelt ID (read only)
segments	integer	Segment set created for contact (read only)
shells	integer	Shell set ID defining structure. Note that if you are creating the seatbelt definition from scratch in JavaScript you **must** define a shell, solid or thick shell set.
slen_1d	real	The initial slack length for any 1D seatbelt elements
slipringFirst	integer	First slipring ID (read only)
slipringLast	integer	Last slipring ID (read only)
solids	integer	Solid set ID defining structure. Note that if you are creating the seatbelt definition from scratch in JavaScript you **must** define a shell, solid or thick shell set.
t1Shell	real	Optional thickness at n1 for any shell elements
t1_2d	real	Optional thickness at n1 for any 2D seatbelt elements
t2Shell	real	Optional thickness at n2 for any shell elements
t2_2d	real	Optional thickness at n2 for any 2D seatbelt elements
t3Shell	real	Optional thickness at n3 for any shell elements
t3_2d	real	Optional thickness at n3 for any 2D seatbelt elements
t4Shell	real	Optional thickness at n4 for any shell elements
t4_2d	real	Optional thickness at n4 for any 2D seatbelt elements
thickFactor	real	Factor used when thickFlag is 1
thickFlag	integer	Thickness used during fitting: 0 (default)=use true thickness; 1=use true thickness x factor; 2=use neutral axis (no thickness)
thickness	real	The thickness of 2D belt elements
title	string	Belt title.
tolerance	real	The convergence tolerance at which fitting halts
tshells	integer	Thick shell set ID defining structure. Note that if you are creating the seatbelt definition from scratch in JavaScript you **must** define a shell, solid or thick shell set.

width	real	The overall belt width
xsectionFirst	integer	First cross section ID (read only)
xsectionLast	integer	Last cross section ID (read only)

Detailed Description

The Belt class allows you to create, modify, and manipulate belt fitting definitions. See the documentation below for more details.

Constructor

new Belt(model*[Model]*, id*[integer]*, title (optional)*[string]*, structural_type (optional)*[string]*, flag (optional)*[integer]*)

Description

Create a new Belt object.

Arguments

Name	Type	Description
model	Model	Model that the belt definition will be created in
id	integer	Belt number
title (optional)	string	Title for the belt
structural_ type (optional)	string	Seatbelt will be fitted around this entity type. This will trigger creation of sets as required. Type can be one of MODEL, DUMMY, PART, any ELEMENT subtype such as SHELL, or any SET subtype such as SET_PART. See Appendix I of the PRIMER manual for more information on PRIMER types
flag (optional)	integer	Flag used to identify entities that the belt should fit around. This argument is ignored if structural_type is MODEL. Instead, the current model is used

Return type

Belt object

Example

To create a new belt called 'Example' in model m with label 100:

```
var b = new Belt(m, 100, 'Example');
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a belt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the belt

Return type

No return value

Example

To associate comment c to the belt b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the belt

Arguments

No arguments

Return type

No return value

Example

To blank belt b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the belts in model m:

```
Belt.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged belts will be blanked in
flag	Flag	Flag set on the belts that you want to blank

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the belts in model m flagged with f:

```
Belt.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the belt is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if belt b is blanked:

```
if (b.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the belt.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the belt

Return type

No return value

Example

To clear flag f for belt b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the belt.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Belt object

Example

To copy belt b into belt z:

```
var z = b.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a belt.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the belt

Return type

No return value

Example

To detach comment c from the belt b:

```
b.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for belt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for belt b:

```
b.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first belt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first belt in

Return type

Belt object (or null if there are no belts in the model).

Example

To get the first belt in model m:

```
var b = Belt.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free belt label in the model. Also see Belt.LastFreeLabel(), Belt.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free belt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Belt label.

Example

To get the first free belt label in model m:

```
var label = Belt.FirstFreeLabel(m);
```

Fit()

Description

(Re)fits belt

Arguments

No arguments

Return type

No return value

Example

To (re)fit belt b:

```
b.Fit();
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the belts in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be flagged in
flag	Flag	Flag to set on the belts

Return type

No return value

Example

To flag all of the belts with flag f in model m:

```
Belt.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the belt is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the belt

Return type

true if flagged, false if not.

Example

To check if belt b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each belt in the model.
Note that ForEach has been designed to make looping over belts as fast as possible and so has some limitations. Firstly, a single temporary Belt object is created and on each function call it is updated with the current belt data. This means that you should not try to store the Belt object for later use (e.g. in an array) as it is temporary. Secondly, you cannot create new belts inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all belts are in
func	function	Function to call for each belt

extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the belts in model m:

```
Belt.ForEach(m, test);
function test(b)
{
// b is Belt object
}
```

To call function test for all of the belts in model m with optional object:

```
var data = { x:0, y:0 };
Belt.ForEach(m, test, data);
function test(b, extra)
{
// b is Belt object
// extra is data
}
```

Generate()

Description

Generates belt mesh. Extracts and uses existing mesh properties when a mesh is present; inserts a default mesh otherwise.

Arguments

No arguments

Return type

No return value

Example

To generate a mesh for belt b:

```
b.Generate();
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of Belt objects for all of the belts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get belts from

Return type

Array of Belt objects

Example

To make an array of Belt objects for all of the belts in model m

```
var b = Belt.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a belt.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the belt b:

```
var comm_array = b.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Belt objects for all of the flagged belts in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get belts from
flag	Flag	Flag set on the belts that you want to retrieve

Return type

Array of Belt objects

Example

To make an array of Belt objects for all of the belts in model m flagged with f

```
var b = Belt.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Belt object for a belt ID.

Arguments

Name	Type	Description
Model	Model	Model to find the belt in
number	integer	number of the belt you want the Belt object for

Return type

Belt object (or null if belt does not exist).

Example

To get the Belt object for belt 100 in model m

```
var b = Belt.GetFromID(m, 100);
```

GetMesh(index*[integer]*)

Description

Returns the information for a belt mesh section (properties base_pt1, base_pt2, path_pt1, path_pt2, mode, lb1, lb2). See Belt.SetMesh() for more information on supported properties. Must be preceded by a call to Belt.Generate().

Arguments

Name	Type	Description
index	integer	The index of the mesh section you want the information for. **Note that mesh segments start at 0, not 1**. 0 <= index < meshSegs

Return type

Object containing the mesh section information

Example

To get the information for the 3rd mesh section for belt b:

```
var info = b.GetMesh(2);
```

GetParameter(prop*[string]*)

Description

Checks if a Belt property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Belt.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	belt property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Belt property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Belt property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

GetPoint(index*[integer]*)

Description

Returns the information for a path point (properties fixity, x, y, z, node, trx1, try1, trz1, tnx1, tny1, tnz1, tnode1, trx2, try2, trz2, tnx2, tny2, tnz2, tnode2). Properties fixity, x, y, z and node will always be returned. Twist properties trx1, try1, trz1, tnx1, tny1, tnz1, tnode1, trx2, try2, trz2, tnx2, tny2, tnz2 and tnode2 will only be returned if defined for the point.

Arguments

Name	Type	Description
index	integer	The index of the path point you want the information for. **Note that path points start at 0, not 1**. 0 <= index < points

Return type

Object containing the path point information

Example

To get the information for the 3rd path point for belt b:

```
var info = b.GetPoint(2);
```

InsertPoint(index*[integer]*, position*[integer]*, data*[object]*)

Description

Inserts a path point before/after an existing one. Subsequent path points will be moved ’up’ as required.

Arguments

Name	Type	Description
index	integer	The index of an existing path point. **Note that path points start at 0, not 1**. 0 <= index < points
position	integer	Do we want to insert before or after the path point denoted by index? The position can be Belt.INSERT_AFTER or Belt.INSERT_BEFORE

<table>
<tr><td>data</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>fixity</td><td>integer</td><td>Point fixity type. Bitwise ’or’ of the Path point fixity constants: Belt.B_POST_SLIPRING, Belt.FREE_SLIPRING, Belt.KNOWN, Belt.RETRACTOR, Belt.TWIST, Belt.XSEC</td></tr>
<tr><td>node (optional)</td><td>integer</td><td>Node label (not required if using x, y and z)</td></tr>
<tr><td>tnode1 (optional)</td><td>integer</td><td>Twist node 1 label</td></tr>
<tr><td>tnode2 (optional)</td><td>integer</td><td>Twist node 2 label</td></tr>
<tr><td>tnx1 (optional)</td><td>real</td><td>X component of normal vector 1</td></tr>
<tr><td>tnx2 (optional)</td><td>real</td><td>X component of normal vector 2</td></tr>
<tr><td>tny1 (optional)</td><td>real</td><td>Y component of normal vector 1</td></tr>
<tr><td>tny2 (optional)</td><td>real</td><td>Y component of normal vector 2</td></tr>
<tr><td>tnz1 (optional)</td><td>real</td><td>Z component of normal vector 1</td></tr>
<tr><td>tnz2 (optional)</td><td>real</td><td>Z component of normal vector 2</td></tr>
<tr><td>trx1 (optional)</td><td>real</td><td>X component of twist radial vector 1</td></tr>
<tr><td>trx2 (optional)</td><td>real</td><td>X component of twist radial vector 2</td></tr>
<tr><td>try1 (optional)</td><td>real</td><td>Y component of twist radial vector 1</td></tr>
<tr><td>try2 (optional)</td><td>real</td><td>Y component of twist radial vector 2</td></tr>
<tr><td>trz1 (optional)</td><td>real</td><td>Z component of twist radial vector 1</td></tr>
<tr><td>trz2 (optional)</td><td>real</td><td>Z component of twist radial vector 2</td></tr>
<tr><td>x (optional)</td><td>real</td><td>X coordinate (not required if using node)</td></tr>
<tr><td>y (optional)</td><td>real</td><td>Y coordinate (not required if using node)</td></tr>
<tr><td>z (optional)</td><td>real</td><td>Z coordinate (not required if using node)</td></tr>
</table>
Object containing the path point data. Object has the following properties:
</td></tr>
</table>

Return type

no return value

Example

To insert a new 2nd path point for belt b with B-Post slipring fixity using twist nodes 1000 and 10001:

```
var data = { fixity:Belt.RETRACTOR|Belt.TWIST, node:999 tnode1:1000, tnode2:1001
};
b.InsertPoint(1, Belt.INSERT_BEFORE, data);
```

Last(Model*[Model]*) [static]

Description

Returns the last belt in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last belt in

Return type

Belt object (or null if there are no belts in the model).

Example

To get the last belt in model m:

```
var b = Belt.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free belt label in the model. Also see Belt.FirstFreeLabel(), Belt.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free belt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Belt label.

Example

To get the last free belt label in model m:

```
var label = Belt.LastFreeLabel(m);
```

Next()

Description

Returns the next belt in the model.

Arguments

No arguments

Return type

Belt object (or null if there are no more belts in the model).

Example

To get the belt in model m after belt b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) belt label in the model. Also see Belt.FirstFreeLabel(), Belt.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free belt label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Belt label.

Example

To get the next free belt label in model m:

```
var label = Belt.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a belt.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only belts from that model can be picked. If the argument is a Flag then only belts that are flagged with *limit* can be selected. If omitted, or null, any belts from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Belt object (or null if not picked)

Example

To pick a belt from model m giving the prompt 'Pick belt from screen':

```
var b = Belt.Pick('Pick belt from screen', m);
```

Previous()

Description

Returns the previous belt in the model.

Arguments

No arguments

Return type

Belt object (or null if there are no more belts in the model).

Example

To get the belt in model m before belt b:

```
var b = b.Previous();
```

RemovePoint(index*[integer]*)

Description

Removes a path point from a belt

Arguments

Name	Type	Description
index	integer	The index of the path point you want to remove. **Note that path points start at 0, not 1**. 0 <= index < points

Return type

no return value

Example

To remove for the 3rd path point for belt b:

```
b.RemovePoint(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the belts in model m, from 1000000:

```
Belt.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged belts will be renumbered in
flag	Flag	Flag set on the belts that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the belts in model m flagged with f, from 1000000:

```
Belt.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select belts using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting belts
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only belts from that model can be selected. If the argument is a Flag then only belts that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any belts can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of belts selected or null if menu cancelled

Example

To select belts from model m, flagging those selected with flag f, giving the prompt 'Select belts':

```
Belt.Select(f, 'Select belts', m);
```

To select belts, flagging those selected with flag f but limiting selection to belts flagged with flag l, giving the prompt 'Select belts':

```
Belt.Select(f, 'Select belts', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the belt.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the belt

Return type

No return value

Example

To set flag f for belt b:

```
b.SetFlag(f);
```

SetMesh(index*[integer]*, data*[object]*)

Description

Sets the data for various properties for a mesh section in a belt. Values for properties not invoked will be retained as is. Must be preceded by a call to Belt.Generate()

Arguments

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>index</td><td>integer</td><td>The index of the mesh section</td></tr>
<tr><td>data</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>base_pt1</td><td>integer</td><td>1st base point number</td></tr>
<tr><td>base_pt2</td><td>integer</td><td>2nd base point number</td></tr>
<tr><td>lb1</td><td>integer</td><td>Number of belt elements at the 1st end for mixed modes</td></tr>
<tr><td>lb2</td><td>integer</td><td>Number of belt elements at the 2nd end for mixed modes</td></tr>
<tr><td>mode</td><td>integer</td><td>Meshing modes can be of old style or new style.
The following old style constants are available: Belt.MSEG_B1_ONLY, Belt.MSEG_B2_ONLY, Belt.MSEG_SH_ONLY, Belt.MSEG_MIX_SB1, Belt.MSEG_MIX_SB2

The following constant must be invoked in order to use the new style: Belt.MSEG_BD_NEW

The following new style constants are available: Belt.MSEG_E1_1D, Belt.MSEG_E1_2D, Belt.MSEG_E1_SH, Belt.MSEG_E2_1D, Belt.MSEG_E2_2D, Belt.MSEG_E2_SH, Belt.MSEG_CE_1D, Belt.MSEG_CE_2D, Belt.MSEG_CE_SH</td></tr>
<tr><td>path_pt1</td><td>integer</td><td>1st path point number</td></tr>
<tr><td>path_pt2</td><td>integer</td><td>2nd path point number</td></tr>
</table>
Object containing the mesh section data. Object has the following properties:
</td></tr>
</table>

Return type

no return value

Example

To set the following properties for the final mesh section: base points: 5, 9, path points: 59, 92, mode: 1D at ends, shells at centre, number of elements at either end: 4 and 10:

```
var data = { base_pt1: 5, base_pt2: 9, path_pt1: 59, path_pt2: 92, mode:
Belt.MSEG_BD_NEW | Belt.MSEG_E1_1D | Belt.MSEG_CE_SH | Belt.MSEG_E2_1D, lb1: 4,
lb2: 10 };
b.SetMesh(b.num_segments-1, data);
```

SetMeshingLabels(entity_type*[constant]*, label_value*[integer]*) [static]

Description

Set the start labels for the entities created for a Seat Belt.

Arguments

Name	Type	Description
entity_type	constant	The Meshing label can be Belt.MESH_NODE, _Belt.MESH_SHELL, Belt.MESH_SET_NODE, Belt.MESH_SET_NODE, Belt.MESH_SEATBELT, Belt.MESH_NRBC, BELT.MESH_RETRACTOR, Belt.MESH_XSEC, Belt.MESH_SLIPRING, Belt.MESH_SET_PART, Belt.MESH_2D_SLIPRING_SET_NODE, Belt.MESH_ALL.
label_value	integer	The intial label value to be assigned for the entity type.

Return type

no return value

Example

To get the initial value of the node label in seatbelt meshing as 1000:

```
BELT.SetMeshingLabels(Belt.MESH_NODE,1000)
```

SetPoint(index*[integer]*, data*[object]*)

Description

Sets the data for a path point in a belt

Arguments

Name	Type	Description
index	integer	The index of the path point you want to set. **Note that path points start at 0, not 1**. To add a new point use index points

<table>
<tr><td>data</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>fixity</td><td>integer</td><td>Point fixity type. Bitwise ’or’ of the Path point fixity constants: Belt.B_POST_SLIPRING, Belt.FREE_SLIPRING, Belt.KNOWN, Belt.RETRACTOR, Belt.TWIST, Belt.XSEC</td></tr>
<tr><td>node (optional)</td><td>integer</td><td>Node label (not required if using x, y and z)</td></tr>
<tr><td>tnode1 (optional)</td><td>integer</td><td>Twist node 1 label</td></tr>
<tr><td>tnode2 (optional)</td><td>integer</td><td>Twist node 2 label</td></tr>
<tr><td>tnx1 (optional)</td><td>real</td><td>X component of normal vector 1</td></tr>
<tr><td>tnx2 (optional)</td><td>real</td><td>X component of normal vector 2</td></tr>
<tr><td>tny1 (optional)</td><td>real</td><td>Y component of normal vector 1</td></tr>
<tr><td>tny2 (optional)</td><td>real</td><td>Y component of normal vector 2</td></tr>
<tr><td>tnz1 (optional)</td><td>real</td><td>Z component of normal vector 1</td></tr>
<tr><td>tnz2 (optional)</td><td>real</td><td>Z component of normal vector 2</td></tr>
<tr><td>trx1 (optional)</td><td>real</td><td>X component of twist radial vector 1</td></tr>
<tr><td>trx2 (optional)</td><td>real</td><td>X component of twist radial vector 2</td></tr>
<tr><td>try1 (optional)</td><td>real</td><td>Y component of twist radial vector 1</td></tr>
<tr><td>try2 (optional)</td><td>real</td><td>Y component of twist radial vector 2</td></tr>
<tr><td>trz1 (optional)</td><td>real</td><td>Z component of twist radial vector 1</td></tr>
<tr><td>trz2 (optional)</td><td>real</td><td>Z component of twist radial vector 2</td></tr>
<tr><td>x (optional)</td><td>real</td><td>X coordinate (not required if using node)</td></tr>
<tr><td>y (optional)</td><td>real</td><td>Y coordinate (not required if using node)</td></tr>
<tr><td>z (optional)</td><td>real</td><td>Z coordinate (not required if using node)</td></tr>
</table>
Object containing the path point data. Object has the following properties:
</td></tr>
</table>

Return type

no return value

Example

To add a new B-Post slipring path point to belt b at node 1000:

```
var data = { fixity:Belt.B_POST_SLIPRING, node:1000 };
b.SetPoint(b.points, data);
```

To add a new path point to belt b at coordinate *10, 20, 30):

```
var data = { fixity:0, x:10, y:20, z:30 };
b.SetPoint(b.points, data);
```

To add a new retractor path point to belt b at (10, 20, 30) with twist nodes 1000 and 1001:

```
var data = { fixity:Belt.RETRACTOR|Belt.TWIST, x:10, y:20, z:30, tnode1:1000,
tnode2:1001 };
b.SetPoint(b.points, data);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the belt. The belt will be sketched until you either call Belt.Unsketch(), Belt.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the belt is sketched. If omitted redraw is true. If you want to sketch several belts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch belt b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged belts in the model. The belts will be sketched until you either call Belt.Unsketch(), Belt.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged belts will be sketched in
flag	Flag	Flag set on the belts that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the belts are sketched. If omitted redraw is true. If you want to sketch flagged belts several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all belts flagged with flag in model m:

```
Belt.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of belts in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing belts should be counted. If false or omitted referenced but undefined belts will also be included in the total.

Return type

number of belts

Example

To get the total number of belts in model m:

```
var total = Belt.Total(m);
```

Unblank()

Description

Unblanks the belt

Arguments

No arguments

Return type

No return value

Example

To unblank belt b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be unblanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the belts in model m:

```
Belt.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged belts will be unblanked in
flag	Flag	Flag set on the belts that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the belts in model m flagged with f:

```
Belt.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all belts will be unset in
flag	Flag	Flag to unset on the belts

Return type

No return value

Example

To unset the flag f on all the belts in model m:

```
Belt.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the belt.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the belt is unsketched. If omitted redraw is true. If you want to unsketch several belts and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch belt b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all belts.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the belts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all belts in model m:

```
Belt.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged belts in the model.

Arguments

Name	Type	Description
Model	Model	Model that all belts will be unsketched in
flag	Flag	Flag set on the belts that you want to unsketch

redraw (optional)	boolean	If model should be redrawn or not after the belts are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all belts flagged with flag in model m:

```
Belt.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Belt object.

Example

To check if Belt property b.example is a parameter by using the Belt.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for belt. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for belt b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this belt.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for belt b:

```
var xrefs = b.Xrefs();
```

Check class

The Check class enables you to access model checking in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- AddDashboardComment(comment*[string]*)
- AddDashboardHealth(model health*[String]*, Health text colour (optional)*[constant]*, Health button colour(optional)*[constant]*)
- Error(message*[string]*, details (optional)*[string]*)
- GetAllDashboards()
- KeyoutHook(interrupt flag*[boolean]*)
- SetDashboardMessage(first*[string]*, second (optional)*[string]*)
- SetDashboardStatus(status*[constant]*)
- Warning(message*[string]*, details (optional)*[string]*)

Check constants

Constants for Dashboard

Name	Description
Check.ERROR	Dashboard check gave error(s)
Check.OK	Dashboard check status OK
Check.UNKNOWN	Dashboard check status unknown (not run)
Check.WARNING	Dashboard check gave warning(s)

Constants for dashboard health colour

Name	Description
Check.BLACK	Colour black
Check.BLUE	Colour blue
Check.CYAN	Colour cyan
Check.DARKBLUE	Colour dark blue
Check.DARKGREEN	Colour dark green
Check.DARKGREY	Colour dark grey
Check.DARKRED	Colour dark red
Check.GREEN	Colour green
Check.GREY	Colour grey
Check.LIGHTGREY	Colour light grey
Check.MAGENTA	Colour magenta
Check.ORANGE	Colour orange

Check.RED	Colour red
Check.WHITE	Colour white
Check.YELLOW	Colour yellow

Detailed Description

The Check class is used add checks to PRIMER using JavaScript. Two different types of checks can be added:

- Individual checks for each node, part, shell etc in a model.
- Custom checks that can reference multiple entities for checking in a model

PRIMER will look in 3 locations for additional JavaScript checks to run when doing checking:

- OA_ADMIN/primer_library/scripts/checks
- OA_INSTALL/primer_library/scripts/checks
- HOME/primer_library/scripts/checks

The directories OA_INSTALL/primer_library/scripts etc can be changed with the primer*script_dir preference.

For individual checks PRIMER will look in these directories for a script with the name 'class_name.js'. For example if you wanted to write a script that will be run for every part in a model the script should be called 'Part.js'.
For custom checks PRIMER will look in these directories for a script called 'custom.js'. This obviously means that there can only be one custom script in each directory. **These filenames are case sensitive**.
Individual scripts will be called with 3 arguments:
arguments[0] = Name of the script
arguments[1] = model object
arguments[2] = Item object
Individual scripts can add warnings or errors by using the Warning() or Error() methods of the appropriate class. For example for a Part the script can call the methods Part.Error() and Part.Warning(). **The script should not call the Error() and Warning() methods of other classes**.
As a simple example of an individual check, suppose you wanted it to be an error if any shell parts in your model did not use type 16 shells. Add a script called 'Part.js' in the directory 'OA_INSTALL/primer_library/scripts/checks' (or one of the other directories) containing:

```
// arguments[0] is name of script
var m = arguments[1]; // arguments[1] is model pointer
var p = arguments[2]; // arguments[2] is part pointer
if (p.exists && p.secid)
{
    var s = Section.GetFromID(m, p.secid);
    if (s.exists && s.type == Section.SHELL && s.elform != 16)
        p.Error("Shell part elform not 16", "Fictional company policy is to use
elform 16 for shell parts");
}
```

Custom scripts will be called with 2 arguments:
arguments[0] = Name of the script
arguments[1] = model object
Custom scripts can add warnings or errors by using the static Check.Error() and Check.Warning() methods. **The script should not call the Error() and Warning() methods of other classes**.
As a simple example of a custom check, suppose a dummy uses node 1000 for the H-point and this should be at coordinates (1000, -500, 100) within tolerance of 0.1 for an analysis . You do not want to run a check for every node in the model (i.e. an individual check). You just want to check that node 1000 is at the correct coordinates. To do this you could create a script called 'custom.js' in the directory 'OA_INSTALL/primer_library/scripts/checks' (or one of the other directories) containing:

```
// arguments[0] is name of script
var m = arguments[1]; // arguments[1] is model pointer
var n = Node.GetFromID(m, 1000);
if (!n)
    Check.Error("No H-point node", "Model does not contain node for dummy
H-point");
if (!n.exists)
    Check.Error("H-point node not defined", "Dummy H-point node is referred to
but not defined");
var dx = n.x - 1000;
var dy = n.y - (-500);
var dy = n.z - 100;
var d = Math.sqrt(dx*dx + dy*dy + dz*dz);
if (d > 0.1)
    Check.Error("H-point not at correct position", "Dummy H-point is "+d+"mm
away from target position");
```

See the documentation below for more details.

Details of functions

AddDashboardComment(comment*[string]*) [static]

Description

Adds a comment for a user dashboard check. Multiple comments can be added. Call this function as many times as required.
This function should only be called from a user JavaScript dashboard script.

Arguments

Name	Type	Description
comment	string	The comment to add.

Return type

No return value

Example

To add a comment:

```
Check.AddDashboardComment("This is a comment");
```

AddDashboardHealth(model health*[String]*, Health text colour (optional)*[constant]*, Health button colour(optional)*[constant]*) [static]

Description

Allows the user to add the value of model health based on the other dashboard results
This function should only be called from model_health.config.js which should be placed with the other user defined dashboard scripts.

Arguments

Name	Type	Description
model health	String	Text which will be displayed on the dashboard panel and the summary files.
Health text colour (optional)	constant	Colour of the model health text. The default colour is Black.
Health button colour(optional)	constant	Colour of the model health button. The default colour is dark grey.

Return type

No return value

Example

To add computed health as "Model Health 85.1%" and the text colour to red and the button colour to green

```
Check.AddDashboardHealth("Model Health 85.1%",Check.RED,Check.GREEN);
```

Error(message*[string]*, details (optional)*[string]*) [static]

Description

Adds a custom error. This function should only be called from a custom JavaScript check script. See the details in the Check class for how to do this.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error":

```
Check.Error('My custom error');
```

GetAllDashboards() [static]

Description

Returns data from all the dashboards that are defined.
This function should only be called from model_health.config.js which should be placed with the other user defined dashboard scripts.
The dashboard properties are:

- result (Overall result of the dashboard)
- title (dashboard title)
- message1 (First message of the dashboard)
- message2 (Second message of the dashboard)
- comments (Array of comments on the dashboard)

Arguments

No arguments

Return type

Array of dashboard objects

Example

To get the status of all the dashboards:

```
Check.GetAllDashboards();
```

For more details on how to use this function, please take a look at the example script model_health.config.js which is present in the dashboard scrips folder

---

KeyoutHook(interrupt flag*[boolean]*) [static]

Description

Used to proceed with or abort the keyout operation (LS-DYNA output) from the keyout_hook.js script. The current hooks are launched just before the keyout operation from the model write tab, writing from the dialogue box and during keyout from the include tree. Please look at the example_keyout_script.js for an example of its usage.

Arguments

Name	Type	Description
interrupt flag	boolean	If this flag is set to true then keyout is aborted else keyout proceeds as usual.

Return type

No return value

Example

To abort a keyout, set the following line in keyout_hook.js:

```
Check.KeyoutHook(true);
```

SetDashboardMessage(first*[string]*, second (optional)*[string]*) [static]

Description

Adds a message for a user dashboard check. Each dashboard can currently show two messages.
This function should only be called from a user JavaScript dashboard script.

Arguments

Name	Type	Description
first	string	The first message to add.
second (optional)	string	The second message to add.

Return type

No return value

Example

To add the message with two lines:

```
Check.SetDashboardMessage("This is a message", "shown on two lines");
```

To add the message with one line:

```
Check.SetDashboardMessage("This is a single message");
```

SetDashboardStatus(status*[constant]*) [static]

Description

Sets the status of a user dashboard check.
This function should only be called from a user JavaScript dashboard script.

Arguments

Name	Type	Description
status	constant	The status. Can be Check.OK, Check.WARNING, Check.ERROR or Check.UNKNOWN.

Return type

No return value

Example

To set the status to OK (green):

```
Check.SetDashboardStatus(Check.OK);
```

Warning(message*[string]*, details (optional)*[string]*) [static]

Description

Adds a custom warning. This function should only be called from a custom JavaScript check script. See the details in the Check class for how to do this.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning":

```
Check.Warning('My custom warning');
```

Colour class

The Colour class contains constants relating to colours. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- GetFromName(name*[string]*)
- RGB(red*[integer]*, green*[integer]*, blue*[integer]*)

Colour constants

Name	Description
Colour.ASSEMBLY	Base colour on assembly
Colour.BACKGROUND	Background colour
Colour.BLACK	Colour black
Colour.BLUE	Colour blue
Colour.CYAN	Colour cyan
Colour.DARK_ORANGE	Colour dark orange
Colour.DEFAULT	Default colour for objects
Colour.GREEN	Colour green
Colour.GREEN_CYAN	Colour green/cyan
Colour.GREY	Colour grey
Colour.INCLUDE	Base colour on include file
Colour.LIGHT_BLUE	Colour light blue
Colour.MAGENTA	Colour magenta
Colour.MATERIAL	For elements with part IDs base colour on material ID
Colour.MEDIUM_BLUE	Colour medium blue
Colour.MODEL	Base colour on model
Colour.NOT_BACKGROUND	Not the background colour
Colour.ORANGE	Colour orange
Colour.PART	For elements with part IDs base colour on part ID
Colour.RED	Colour red
Colour.RED_MAGENTA	Colour red/magenta
Colour.SECTION	For elements with part IDs base colour on section ID
Colour.SKETCH	Sketch colour
Colour.TEXT	Text colour

Colour.WHITE	Colour white
Colour.YELLOW	Colour yellow
Colour.YELLOW_GREEN	Colour yellow/green

Detailed Description

The Colour class is used to define colours, either by predefined colours or by RGB values. The easiest way to set the colour of something is to use the predefined colour constants. e.g. to set the colour of part p to red:

```
p.colour = Colour.RED;
```

For other colours use Colour.RGB().

Details of functions

GetFromName(name*[string]*) [static]

Description

Returns the colour for a given core or user colour name

Arguments

Name	Type	Description
name	string	The name of the colour, for example red or user_green.

Return type

colour value (integer)

RGB(red*[integer]*, green*[integer]*, blue*[integer]*) [static]

Description

Creates a colour from red, green and blue components

Arguments

Name	Type	Description
red	integer	red component of colour (0-255).
green	integer	green component of colour (0-255).
blue	integer	blue component of colour (0-255).

Return type

colour value (integer)

Example

To set the colour of model m to red:

```
m.SetColour( Colour.RGB(255, 0, 0) );
```

To set the colour of part p to red:

```
p.colour = Colour.RGB(255, 0, 0);
```

Conx class

The Conx class gives you access to connections in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RealizeAll(Model*[Model]*)
- RealizeFlagged(Model*[Model]*, flag*[Flag]*)
- ReloadConnectors()
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SetRuleDiameter(diameter*[integer]*)
- SetRuleFEPID(pid*[integer]*)
- SetRulePID(pid*[integer]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UseSPR2Pref(option*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- EmptyPatch()
- Error(message*[string]*, details (optional)*[string]*)
- ExtractColour()
- Flagged(flag*[Flag]*)
- GetComments()
- GetElements()
- GetEntities(type*[string]*)
- GetLayerData(layer*[integer]*)
- GetLayerShells(layer*[integer]*)
- GetParameter(prop*[string]*)
- GetPatchCoords(point*[integer]*)
- GetPatchTopol(point*[integer]*)

- GetPathData(point*[integer]*)
- GetPidData()
- GetSettings()
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemovePatchTopol(layer*[integer]*)
- RemovePathData(layer*[integer]*)
- SetFlag(flag*[Flag]*)
- SetLayerData(layer*[integer]*, item1*[integer/string]*, item2 (optional)*[integer/string]*, ... (optional)*[integer/string]*)
- SetPatchCoords(point*[integer]*, x*[real]*, y*[real]*, z*[real]*)
- SetPatchTopol(point*[integer]*, c1*[integer]*, c2*[integer]*, c3*[integer]*, c4 (optional)*[integer]*)
- SetPathData(point*[integer]*, x*[real]*, y*[real]*, z*[real]*)
- SetPidData(item1*[integer/string]*, item2 (optional)*[integer/string]*, ... (optional)*[integer/string]*)
- SetSettings(data*[object]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Conx constants

Name	Description
Conx.ADHESIVE	Connection is adhesive.
Conx.ADHESIVE_PATCH	Connection adhesive type is a patch.
Conx.ADHESIVE_SOLID	Connection adhesive type is a solid line.
Conx.ASSEMBLY	If the connection refers to an assembly rather than individual layers, the assembly is defined by part tree assembly.
Conx.BAD	Connection is bad (e.g. necessary data is missing).
Conx.BOLT	Connection is a bolt.
Conx.BOLT_MODULE	Library bolt.
Conx.BOLT_MRG_2PTS	2pt Patch Beam.
Conx.BOLT_MRG_2PTS_RB	2pt Patch (Rigid Beam).
Conx.BOLT_MRG_2PTS_RJ	2pt Patch Revolute joint.
Conx.BOLT_MRG_CYL	Cylindrical Merge.
Conx.BOLT_MRG_CYL_BALL	Cylindrical Patch Ball joint.
Conx.BOLT_MRG_CYL_BEAM	Cylindrical Patch Beam.
Conx.BOLT_NRB_2PTS	2pt NRB Beam.
Conx.BOLT_NRB_CYL	Cylindrical NRB.
Conx.BOLT_NRB_CYL_BALL	Cylindrical NRB Ball joint.
Conx.BOLT_NRB_CYL_BEAM	Cylindrical NRB Beam.

Conx.BOLT_NRB_SPH	Spherical NRB.
Conx.BOLT_NRB_SPH_BALL	Spherical NRB Ball joint.
Conx.BOLT_NRB_SPH_DISC	Spherical NRB Discrete Beam.
Conx.DORMANT	Connection is dormant (not yet made).
Conx.INVALID	Connection has been made but something is wrong (e.g. part moved).
Conx.MADE	Connection has been made but status is unknown.
Conx.PART_SET	If the connection refers to an assembly rather than individual layers, the assembly is defined by part set.
Conx.REALIZED	Connection has been made and is OK (checks OK).
Conx.RIGID	This constant is deprecated in version 10.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Conx.BOLT instead. [deprecated]
Conx.RIGID_MERGE	This constant is deprecated in version 10.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Conx.BOLT_MRG_CYL instead. [deprecated]
Conx.RIGID_NRB	This constant is deprecated in version 10.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions. Please use Conx.BOLT_NRB_CYL instead. [deprecated]
Conx.RIVET	Connection is rivet.
Conx.SPOTWELD	Connection is a spotweld.
Conx.SPOTWELD_BEAM	Connection spotweld type is beam.
Conx.SPOTWELD_LINE	Connection is a spotweld line.
Conx.SPOTWELD_MIG	Connection spotweld type is (beam) MIG weld.
Conx.SPOTWELD_SOLID1	Connection spotweld type is one solid/spotweld layer.
Conx.SPOTWELD_SOLID12	Connection spotweld type is twelve solids/spotweld layer.
Conx.SPOTWELD_SOLID16	Connection spotweld type is sixteen solids/spotweld layer.
Conx.SPOTWELD_SOLID4	Connection spotweld type is four solids/spotweld layer.
Conx.SPOTWELD_SOLID8	Connection spotweld type is eight solids/spotweld layer.

Conx properties

Name	Type	Description
adhesive_esize	real	Element size along the length of the adhesive run
adhesive_nelem	integer	The number of elements across the width of the adhesive
adhesive_width	real	The width of the adhesive run

angtol	real	angle tolerance for bolt
angtol2	real	angle tolerance at end 2 for 2 point bolt
assembly	integer/string	Assembly used to specify panels connection together, rather than individual layers. Integer for a part set ID, string for a Primer assembly (name).
assembly_ type	constant	The assembly type. Can be Conx.PART_SET or Conx.ASSEMBLY.
colour	Colour	The colour of the connection
diameter	real	Diameter of spotweld/rigid
diameter2	real	Diameter of rigid at end 2
edge_ distance	real	Spotweld line edge distance
edge_lock	logical	true if a spotweld line is locked to an edge, false if not
error	string	Description of the error if the connection cannot be made (read only)
error_details	string	Details of the error if the connection cannot be made (read only)
fit	integer	contact fitting method for library bolts
id	integer	Conx number. Also see the label property which is an alternative name for this.
include	integer	The Include file number that the connection is in.
label	integer	Conx number. Also see the id property which is an alternative name for this.
layers	integer	The number of layers the connection has.
length	real	Length of 1 point bolt, max thickness for 2 point bolt
length2	real	max thickness at end 2 for 2 point bolt
material	integer	The ID of the Material used for 'merge' bolt connections. i.e. Conx.BOLT_MRG_CYL, Conx.BOLT_MRG_CYL_BEAM,
model	integer	The Model number that the connection is in.
module	string	name of library module for bolt
part	integer	The ID of the Part used for adhesive or spotweld connections. Note that in v11.0 and above you are able to specify a different part IDs for elements in the connection between different layers. If you only have one part for the elements in the connection, then this is the value of this property. If there is more than one used, then the value of this property is the first part. If you set this property to a new value, then the all the elements in the connection will have this new part ID when it is realized. To set and retrieve information on parts used between different layers, the functions GetPidData() and SetPidData() should be used.
patch_ coords	integer	The number of patch coordinate points the connection has (Adhesive patch only).
patch_topol	integer	The number of patch topology entries the connection has (Adhesive patch only).
path	integer	The number of path points the connection has (Adhesive only). Note that these points do **NOT** include the start and end points for the adhesive run. These are defined using the properties x, y, z and x2, y2, z2
pitch	real	Spotweld line pitch
resize	integer	snap to points fitting method for library bolts
saved_ settings	boolean	Whether settings are saved for a connection or not
shape	integer	shape for bolt attachment
shape2	integer	shape for bolt attachment at end 2 for 2 point bolt

spr2_id	integer	Internal label of C_SPR2 which applied to this rivet connection (read only)
spr2_match	boolean	True to use matching C_SPR2 for this rivet. False to create new C_SPR2 for each rivet. IF unset, a new C_SPR2 will be created.
spr2_ unshared	boolean	True if C_SPR2 is unique for this rivet (read only)
status	constant	The status of the connection. (read only). Can be Conx.DORMANT, Conx.MADE, Conx.INVALID, Conx.REALIZED or Conx.BAD.
subtype	constant	The connection subtype. For SPOTWELD and SPOTWELD_LINE connections the subtype can be: • Conx.SPOTWELD_BEAM • Conx.SPOTWELD_MIG • Conx.SPOTWELD_SOLID1 • Conx.SPOTWELD_SOLID4 • Conx.SPOTWELD_SOLID8 • Conx.SPOTWELD_SOLID12 • Conx.SPOTWELD_SOLID16 For BOLT connections the subtype can be: • Conx.BOLT_MRG_CYL • Conx.BOLT_MRG_CYL_BEAM • Conx.BOLT_MRG_CYL_BALL • Conx.BOLT_MRG_2PTS • Conx.BOLT_MRG_2PTS_RB • Conx.BOLT_MRG_2PTS_RJ • Conx.BOLT_MRG_CYL • Conx.BOLT_NRB_CYL_BEAM • Conx.BOLT_NRB_CYL_BALL • Conx.BOLT_NRB_SPH • Conx.BOLT_NRB_SPH_BALL • Conx.BOLT_NRB_SPH_DISC • Conx.BOLT_NRB_2PTS • Conx.BOLT_MODULE For ADHESIVE connections the subtype can be: Conx.ADHESIVE_SOLID. Conx.ADHESIVE_PATCH.
title	string	Title for connection
transparency	integer	The transparency of the connection (0-100) 0% is opaque, 100% is transparent.
type	constant	The connection type. Can be Conx.SPOTWELD, Conx.BOLT or Conx.RIVET or Conx.ADHESIVE.
user_data	string	User data for connection
x	real	X coordinate
x2	real	X coordinate for second point (adhesive only)
y	real	Y coordinate
y2	real	Y coordinate for second point (adhesive only)
z	real	Z coordinate
z2	real	Z coordinate for second point (adhesive only)

Detailed Description

The Conx class allows you to create, modify, edit and manipulate connections. See the documentation below for more details.

Constructor

new Conx(Model*[Model]*, x*[real]*, y*[real]*, z*[real]*, type (optional)*[constant]*, subtype (optional)*[constant]*, title (optional)*[string]*)

Description

Create a new Conx object.

Arguments

Name	Type	Description
Model	Model	Model that connection will be created in
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate
type (optional)	constant	Type of connection. Can be Conx.SPOTWELD, Conx.BOLT, Conx.ADHESIVE Conx.SPOTWELD_LINE or Conx.RIVET. If omitted type will be set to Conx.SPOTWELD.
subtype (optional)	constant	Subtype of connection. See property subtype for valid values. If omitted subtype will be set to the default subtype for this type of connection.
title (optional)	string	Title for the connection

Return type

Conx object

Example

To create a new connection in model m, at coordinates (20, 40, 10)

```
var c = new Conx(m, 20, 40, 10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a connection.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the connection

Return type

No return value

Example

To associate comment c to the connection c:

```
c.AssociateComment(c);
```

Blank()

Description

Blanks the connection

Arguments

No arguments

Return type

No return value

Example

To blank connection c:

```
c.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the connections in model m:

```
Conx.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged connections will be blanked in
flag	Flag	Flag set on the connections that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the connections in model m flagged with f:

```
Conx.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the connection is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if connection c is blanked:

```
if (c.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the connection.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the connection

Return type

No return value

Example

To clear flag f for connection c:

```
c.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the connection.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Conx object

Example

To copy connection c into connection z:

```
var z = c.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a connection.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the connection

Return type

No return value

Example

To detach comment c from the connection c:

```
c.DetachComment(c);
```

EmptyPatch()

Description

Empties the patch topology/coordinates data.

Arguments

No arguments

Return type

No return value.

Example

To empty the patch topology/coordinates data for connection c;

```
c.EmptyPatch();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for connection. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for connection c:

```
c.Error("My custom error");
```

ExtractColour()

Description

Extracts the **actual** colour used for connection.
By default in PRIMER many entities such as elements get their colour automatically from the part that they are in. PRIMER cycles through 13 default colours based on the label of the entity. In this case the connection colour property will return the value Colour.PART instead of the actual colour. This method will return the actual colour which is used for drawing the connection.

Arguments

No arguments

Return type

colour value (integer)

Example

To return the colour used for drawing connection c:

```
var colour = c.ExtractColour();
```

First(Model*[Model]*) [static]

Description

Returns the first connection in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first connection in

Return type

Conx object (or null if there are no connections in the model).

Example

To get the first connection in model m:

```
var c = Conx.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free connection label in the model. Also see Conx.LastFreeLabel(), Conx.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free connection label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Conx label.

Example

To get the first free connection label in model m:

```
var label = Conx.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the connections in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be flagged in
flag	Flag	Flag to set on the connections

Return type

No return value

Example

To flag all of the connections with flag f in model m:

```
Conx.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the connection is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the connection

Return type

true if flagged, false if not.

Example

To check if connection c has flag f set on it:

```
if (c.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each connection in the model.
Note that ForEach has been designed to make looping over connections as fast as possible and so has some limitations.
Firstly, a single temporary Conx object is created and on each function call it is updated with the current connection data. This means that you should not try to store the Conx object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new connections inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all connections are in
func	function	Function to call for each connection
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the connections in model m:

```
Conx.ForEach(m, test);
function test(c)
{
// c is Conx object
}
```

To call function test for all of the connections in model m with optional object:

```
var data = { x:0, y:0 };
Conx.ForEach(m, test, data);
function test(c, extra)
{
// c is Conx object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Conx objects for all of the connections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get connections from

Return type

Array of Conx objects

Example

To make an array of Conx objects for all of the connections in model m

```
var c = Conx.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a connection.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the connection c:

```
var comm_array = c.GetComments();
```

GetElements()

Description

Returns the beams/solids that are used in the connection weld.

Arguments

No arguments

Return type

An array containing the element IDs (or null if no elements).

Example

To get the elements for connection c:

```
var elems = c.GetElements();
```

GetEntities(type*[string]*)

Description

Returns list of the entities of type that are used in the connection.

Arguments

Name	Type	Description
type	string	The type of the item in the reference list (for a list of types see Appendix I of the PRIMER manual).

Return type

An array containing the item IDs (or null if none).

Example

To get list of nodes for connection c:

```
var items = c.GetEntities("NODE");
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Conx objects for all of the flagged connections in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get connections from
flag	Flag	Flag set on the connections that you want to retrieve

Return type

Array of Conx objects

Example

To make an array of Conx objects for all of the connections in model m flagged with f

```
var c = Conx.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Conx object for a connection ID.

Arguments

Name	Type	Description
Model	Model	Model to find the connection in
number	integer	number of the connection you want the Conx object for

Return type

Conx object (or null if connection does not exist).

Example

To get the Conx object for connection 100 in model m

```
var c = Conx.GetFromID(m, 100);
```

GetLayerData(layer*[integer]*)

Description

Returns the data for a layer of the connection.

Arguments

Name	Type	Description
layer	integer	The layer you want the data for. **Note that layers start at 0, not 1**.

Return type

An array containing the layer data.

Example

To get the data for the 3rd layer for connection c:

```
var l_data = c.GetLayerData(2);
```

GetLayerShells(layer*[integer]*)

Description

Returns the attached shells for a layer of the connection.

Arguments

Name	Type	Description
layer	integer	The layer you want the data for. **Note that layers start at 0, not 1**.

Return type

Array of Shell objects or null if not valid

Example

To get the attached shells for the 3rd layer for connection c:

```
var shells = c.GetLayerShells(2);
```

GetParameter(prop*[string]*)

Description

Checks if a Conx property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Conx.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	connection property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Conx property c.example is a parameter:

```
Options.property_parameter_names = true;
if (c.GetParameter(c.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Conx property c.example is a parameter by using the GetParameter method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

GetPatchCoords(point*[integer]*)

Description

Returns the data for a patch coordinate of an adhesive patch connection.

Arguments

Name	Type	Description
point	integer	The point you want the data for. **Note that points start at 0, not 1**.

Return type

An array containing the patch coordinate.

Example

To get the data for the 3rd patch coordinate for connection c:

```
var p_data = c.GetPatchCoords(2);
```

GetPatchTopol(point*[integer]*)

Description

Returns the topology for a patch quad/tria of an adhesive patch connection.

Arguments

Name	Type	Description
point	integer	The patch quad/tria you want the data for. **Note that points start at 0, not 1**.

Return type

Array of numbers containing the patch topology information.

Example

To get the data for the 3rd patch quad/tria for connection c:

```
var p_data = c.GetPatchTopol(2);
```

GetPathData(point*[integer]*)

Description

Returns the data for a path point of an adhesive/spotweld line connection.

Arguments

Name	Type	Description
point	integer	The point you want the data for. **Note that points start at 0, not 1**.

Return type

An array containing the path data.

Example

To get the data for the 3rd path point for connection c:

```
var p_data = c.GetPathData(2);
```

GetPidData()

Description

Returns an array of Part objects for the connection FE entities. A connection can contain elements with different part ID's between different layers. If one part ID is returned, that part is used for all elements in the connection. Not applicable for bolts.

Arguments

No arguments

Return type

Array of Part objects

Example

To make an array of Part objects for connection c

```
var arr = Conx.GetPidData();
```

GetSettings()

Description

Returns an object of settings stored with the connection.

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
angle_tolerance	real	Angle tolerance
bolt_adjust_mass	boolean	Adjust bolt mass when creating bolt entitites
bolt_dth_beam	boolean	Add database history beam when bolt beam is created
bolt_feature_line	boolean	Consider feature line for bolt holes
bolt_nrb_min_mass	real	Bolt rigid NRB minimum mass
bolt_part_min_mass	real	Bolt rigid part minimum mass
clinch	boolean	Allow connections to join a clinch type connection
edge_distance	real	Edge distance
glue_break_angle	real	Glue break angle
glue_hard_aspect	real	Glue hard aspect ratio
glue_soft_aspect	real	Glue soft aspect ratio

length_check	boolean	Check length
max_length	real	Maximum length
max_panels	integer	Maximum number of panels
max_warpage	real	Maximum warpage
min_length	real	Minimum length
panel_check	boolean	Check for maximum number of panels
patch_angle	real	Patch angle setting
patch_angle_check	boolean	Turn on or off patch angle check
same_part	boolean	Allow connections to join a part to itself
solid_free_edges	boolean	Consider free edges when orienting single solid spotwelds
spot_line_tol	real	Spotweld line search tolerance
spot_thickness	real	Search thickness
total_length	real	Total length
use_pid	boolean	Use _PID for beam connections
warpage_check	boolean	Check warpage value

Example

To make an Oject containing the stored settings of connection c

```
var o = Conx.GetSettings();
```

Keyword()

Description

Returns the keyword for this connection (*CONNECTION_START_SPOTWELD etc). **Note that a carriage return is not added**. See also Conx.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for connection c:

```
var key = c.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the connection. **Note that a carriage return is not added**. See also Conx.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for connection c:

```
var cards = c.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last connection in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last connection in

Return type

Conx object (or null if there are no connections in the model).

Example

To get the last connection in model m:

```
var c = Conx.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free connection label in the model. Also see Conx.FirstFreeLabel(), Conx.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free connection label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Conx label.

Example

To get the last free connection label in model m:

```
var label = Conx.LastFreeLabel(m);
```

Next()

Description

Returns the next connection in the model.

Arguments

No arguments

Return type

Conx object (or null if there are no more connections in the model).

Example

To get the connection in model m after connection c:

```
var c = c.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) connection label in the model. Also see Conx.FirstFreeLabel(), Conx.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free connection label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Conx label.

Example

To get the next free connection label in model m:

```
var label = Conx.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a connection.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only connections from that model can be picked. If the argument is a Flag then only connections that are flagged with *limit* can be selected. If omitted, or null, any connections from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Conx object (or null if not picked)

Example

To pick a connection from model m giving the prompt 'Pick connection from screen':

```
var c = Conx.Pick('Pick connection from screen', m);
```

Previous()

Description

Returns the previous connection in the model.

Arguments

No arguments

Return type

Conx object (or null if there are no more connections in the model).

Example

To get the connection in model m before connection c:

```
var c = c.Previous();
```

RealizeAll(Model[*Model*]) [static]

Description

Realizes all of the connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be realized in

Return type

No return value

Example

To realize all of the connections in model m:

```
Conx.RealizeAll(m);
```

RealizeFlagged(Model[*Model*], flag[*Flag*]) [static]

Description

Realizes all of the flagged connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged connections will be realized in
flag	Flag	Flag set on the connections that you want to realize

Return type

No return value

Example

To realize all of the connections in model m flagged with f:

```
Conx.RealizeFlagged(m, f);
```

ReloadConnectors() [static]

Description

Reload all modules from primer_library/connectors

Arguments

No arguments

Return type

No return value

Example

```
Conx.ReloadConnectors();
```

RemovePatchTopol(layer*[integer]*)

Description

Deletes the topology at a particular location for patch type adhesive.

Arguments

Name	Type	Description
layer	integer	The topology location you want to remove. **Note that layers start at 0, not 1**.

Return type

No return value.

Example

To remove the 3rd topology data for connection c:

```
c.RemovePatchTopol(2);
```

RemovePathData(layer*[integer]*)

Description

Deletes a pathc point for a line adhesive connection.

Arguments

Name	Type	Description
layer	integer	The point you want to remove. **Note that layers start at 0, not 1**.

Return type

No return value.

Example

To remove the 3rd point from connection c:

```
c.RemovePathData(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the connections in model m, from 1000000:

```
Conx.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged connections will be renumbered in
flag	Flag	Flag set on the connections that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the connections in model m flagged with f, from 1000000:

```
Conx.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select connections using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting connections
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only connections from that model can be selected. If the argument is a Flag then only connections that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any connections can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of connections selected or null if menu cancelled

Example

To select connections from model m, flagging those selected with flag f, giving the prompt 'Select connections':

```
Conx.Select(f, 'Select connections', m);
```

To select connections, flagging those selected with flag f but limiting selection to connections flagged with flag l, giving the prompt 'Select connections':

```
Conx.Select(f, 'Select connections', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the connection.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the connection

Return type

No return value

Example

To set flag f for connection c:

```
c.SetFlag(f);
```

SetLayerData(layer*[integer]*, item1*[integer/string]*, item2 (optional)*[integer/string]*, ... (optional)*[integer/string]*)

Description

Sets the data for a layer of the connection.

Arguments

Name	Type	Description
layer	integer	The layer you want to set the data for. **Note that layers start at 0, not 1**.

item1	integer/string	The first item for the layer definition. As layer definitions can be part IDs, part names, CAD names, part set IDs, part set names or assemby names the following logic is used. If the item is an integer it is assumed to be a part ID. If the item is a string then it must be in the format 'P<part ID>', 'P:<part name>', 'C:<CAD name>', 'S<set ID>', 'S:<set name>' or 'A:<assembly name>'.
item2 (optional)	integer/string	The second item for the layer definition. **This must be type same type as item1. e.g. if item1 is a part ID, item2 must be a part ID (it cannot be a part name etc).**
... (optional)	integer/string	The nth item for the layer definition. **This must be type same type as item1. e.g. if item1 is a part ID, this item must be a part ID (it cannot be a part name etc).**

Return type

No return value.

Example

To set the data for the 3rd layer for connection c, to be part IDs 10 and 20:

```
c.SetLayerData(2, 10, 20);
```

or

```
var a = new Array(10, 20);
c.SetLayerData(2, a);
```

SetPatchCoords(point*[integer]*, x*[real]*, y*[real]*, z*[real]*)

Description

Sets a coordinate used by the adhesive patch connection type.

Arguments

Name	Type	Description
point	integer	The point you want to set the data for. **Note that points start at 0, not 1**.
x	real	X coordinate of point
y	real	Y coordinate of point
z	real	Z coordinate of point

Return type

No return value.

Example

To set the position for the 3rd patch point for connection c, to be (10, 20, 30);

```
c.SetPatchCoords(2, 10, 20, 30);
```

SetPatchTopol(point*[integer]*, c1*[integer]*, c2*[integer]*, c3*[integer]*, c4 (optional)*[integer]*)

Description

Sets the topology used by the adhesive patch connection type.

Arguments

Name	Type	Description
point	integer	The point you want to set the data for. **Note that points start at 0, not 1**.

c1	integer	1st coordinate location point
c2	integer	2nd coordinate location point
c3	integer	3rd coordinate location point
c4 (optional)	integer	4th coordinate location point

Return type

No return value.

Example

To set the topology for the 3rd patch quad/tria for connection c, to be (1, 4, 3, 6);

```
c.SetPatchTopol(2, 1, 4, 3, 6);
```

SetPathData(point*[integer]*, x*[real]*, y*[real]*, z*[real]*)

Description

Sets the data for a path point of the connection.

Arguments

Name	Type	Description
point	integer	The point you want to set the data for. **Note that points start at 0, not 1**.
x	real	X coordinate of point
y	real	Y coordinate of point
z	real	Z coordinate of point

Return type

No return value.

Example

To set the position for the 3rd path point for connection c, to be (10, 20, 30);

```
c.SetPathData(2, 10, 20, 30);
```

SetPidData(item1*[integer/string]*, item2 (optional)*[integer/string]*, ... (optional)*[integer/string]*)

Description

Sets the element part IDs for the connection. A different part can be defined for elements in the connection between different layers. Not applicable for bolts.

Arguments

Name	Type	Description
item1	integer/string	Part label of the first item in the PID layer list.
item2 (optional)	integer/string	The second item for the layer definition.
... (optional)	integer/string	The nth item for the layer definition.

Return type

No return value.

Example

To set the part data for c, to be part IDs 10 and 20:

```
c.SetPidData(10, 20);
```

or

```
var a = new Array(10, 20);
c.SetPidData(a);
```

SetRuleDiameter(diameter*[integer]*) [static]

Description

Set the diameter for a spotweld ring when running a rule. Note that this method can only be called when running a connection rule script. It will not have any effect if used in a 'normal' script.

Arguments

Name	Type	Description
diameter	integer	The diameter to set for the ring

Return type

No return value

Example

To set the diameter for a ring to be 10.0:

```
Conx.SetRuleDiameter(10.0);
```

SetRuleFEPID(pid*[integer]*) [static]

Description

Set the PID for spotweld beam/solid elements or adhesive solids when running a rule. Note that this method can only be called when running a connection rule script. It will not have any effect if used in a 'normal' script.

Arguments

Name	Type	Description
pid	integer	The PID to set for the spotweld or adhesive elements

Return type

No return value

Example

To set the PID for a spotweld to be 1000:

```
Conx.SetRuleFEPID(1000);
```

SetRulePID(pid*[integer]*) [static]

Description

Set the PID for a spotweld ring when running a rule. Note that this method can only be called when running a connection rule script. It will not have any effect if used in a 'normal' script.

Arguments

Name	Type	Description
pid	integer	The PID to set for the ring

Return type

No return value

Example

To set the PID for a ring to be 1000:

```
Conx.SetRulePID(1000);
```

SetSettings(data*[object]*)

Description

Sets the settings stored on a connection entity. Not applicable for bolts.

Arguments

Name	Type	Description

data | object | Object containing the connection settings data. The properties can be: Object has the following properties:

Name	Type	Description
angle_tolerance (optional)	real	Angle tolerance
bolt_adjust_mass (optional)	boolean	Adjust bolt mass when creating bolt entities
bolt_dth_beam (optional)	boolean	Add database history beam when bolt is created
bolt_feature_line (optional)	boolean	Consider feature line for bolt holes
bolt_nrb_min_mass (optional)	real	Bolt NRB minimum mass
bolt_part_min_mass (optional)	real	Bolt rigid part minimum mass
clinch (optional)	boolean	Allow connections to join a clinch type connection
edge_distance (optional)	real	Edge distance
glue_break_angle (optional)	real	Glue break angle
glue_hard_aspect (optional)	real	Glue hard aspect ratio
glue_soft_aspect (optional)	real	Glue soft aspect ratio
length_check (optional)	boolean	Check the connection length
max_length (optional)	real	Maximum length of connection
max_panels (optional)	integer	Maximum number of panels
max_warpage (optional)	real	Maximum warpage
min_length (optional)	real	Minimum length of connection
panel_check (optional)	boolean	Check for maximum number of panels
patch_angle (optional)	real	Patch angle
patch_angle_check (optional)	boolean	Check the patch angle
same_part (optional)	boolean	Allow connections to join a part to itself
solid_free_edges (optional)	boolean	Consider free edges when orienting single solid spotwelds
spot_line_tol (optional)	real	Spotweld line search tolerance
spot_thickness (optional)	real	Search thickness
total_length (optional)	real	Total length of connection
use_pid (optional)	boolean	Use _PID for beam connections
warpage_check (optional)	boolean	Check warpage value

Return type

No return value.

Example

To set the various settings for a connection c:

```
var data = { length_check:true, total_length:1.5, warpage_check:false, angle_
tolerance:5.0 };
c.SetSettings(data);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the connection. The connection will be sketched until you either call Conx.Unsketch(), Conx.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the connection is sketched. If omitted redraw is true. If you want to sketch several connections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch connection c:

```
c.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged connections in the model. The connections will be sketched until you either call Conx.Unsketch(), Conx.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged connections will be sketched in
flag	Flag	Flag set on the connections that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the connections are sketched. If omitted redraw is true. If you want to sketch flagged connections several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all connections flagged with flag in model m:

```
Conx.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of connections in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing connections should be counted. If false or omitted referenced but undefined connections will also be included in the total.

Return type

number of connections

Example

To get the total number of connections in model m:

```
var total = Conx.Total(m);
```

Unblank()

Description

Unblanks the connection

Arguments

No arguments

Return type

No return value

Example

To unblank connection c:

```
c.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the connections in model m:

```
Conx.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged connections will be unblanked in
flag	Flag	Flag set on the connections that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the connections in model m flagged with f:

```
Conx.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all connections will be unset in
flag	Flag	Flag to unset on the connections

Return type

No return value

Example

To unset the flag f on all the connections in model m:

```
Conx.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the connection.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the connection is unsketched. If omitted redraw is true. If you want to unsketch several connections and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch connection c:

```
c.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all connections.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the connections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all connections in model m:

```
Conx.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged connections in the model.

Arguments

Name	Type	Description
Model	Model	Model that all connections will be unsketched in
flag	Flag	Flag set on the connections that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the connections are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all connections flagged with flag in model m:

```
Conx.UnsketchAll(m, flag);
```

UseSPR2Pref(option*[boolean]*) [static]

Description

Default true means use the pref settings for C_SPR2 created when rivet realized.

Arguments

Name	Type	Description

option	boolean	Default true means use the pref settings for C_SPR2 created when rivet realized.

Return type

No return value

Example

To ignore any pref settings and use zero for newly created C_SPR2 cards

```
Conx.UseSPR2Pref(false);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Conx object.

Example

To check if Conx property c.example is a parameter by using the Conx.GetParameter() method:

```
if (c.ViewParameters().GetParameter(c.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for connection. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for connection c:

```
c.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this connection.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for connection c:

```
var xrefs = c.Xrefs();
```

---

toString()

Description

Creates a string containing the connection data in keyword format. Note that this contains the keyword header and the keyword cards. See also Conx.Keyword() and Conx.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for connection n in keyword format

```
var s = c.toString();
```

Dummy class

The Dummy class gives you access to dummy cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetAssembly(index*[integer]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetPoint(index*[integer]*)
- GetPointData(rpt*[integer]*)
- GetPointTitle(rpt*[integer]*)
- Next()
- Previous()
- RemovePoint(index*[integer]*)
- SetFlag(flag*[Flag]*)
- SetPoint(index*[integer]*, data*[object]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

Dummy properties

Name	Type	Description
assemblies	integer	Number of assemblies defined. (read only)
exists	logical	true if dummy exists, false if referred to but not defined. (read only)
id	integer	Dummy number. Also see the label property which is an alternative name for this. (read only)
include	integer	The Include file number that the dummy is in.
label	integer	Dummy number. Also see the id property which is an alternative name for this. (read only)
model	integer	The Model number that the dummy is in.
points	integer	Number of reference points defined. (read only)
title	string	Dummy title.
xhpoint	real	H-Point X coordinate. (read only)
yhpoint	real	H-Point Y coordinate. (read only)
zhpoint	real	H-Point Z coordinate. (read only)

Detailed Description

The Dummy class allows you to create, modify, edit and manipulate dummy cards. See the documentation below for more details.

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a dummy.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the dummy

Return type

No return value

Example

To associate comment c to the dummy d:

```
d.AssociateComment(c);
```

Blank()

Description

Blanks the dummy

Arguments

No arguments

Return type

No return value

Example

To blank dummy d:

```
d.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the dummies in model m:

```
Dummy.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged dummies will be blanked in
flag	Flag	Flag set on the dummies that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the dummies in model m flagged with f:

```
Dummy.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the dummy is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if dummy d is blanked:

```
if (d.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the dummy.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the dummy

Return type

No return value

Example

To clear flag f for dummy d:

```
d.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the dummy.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Dummy object

Example

To copy dummy d into dummy z:

```
var z = d.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a dummy.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the dummy

Return type

No return value

Example

To detach comment c from the dummy d:

```
d.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for dummy. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for dummy d:

```
d.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first dummy in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first dummy in

Return type

Dummy object (or null if there are no dummies in the model).

Example

To get the first dummy in model m:

```
var d = Dummy.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free dummy label in the model. Also see Dummy.LastFreeLabel(), Dummy.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free dummy label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Dummy label.

Example

To get the first free dummy label in model m:

```
var label = Dummy.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the dummies in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be flagged in
flag	Flag	Flag to set on the dummies

Return type

No return value

Example

To flag all of the dummies with flag f in model m:

```
Dummy.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the dummy is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the dummy

Return type

true if flagged, false if not.

Example

To check if dummy d has flag f set on it:

```
if (d.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each dummy in the model.
Note that ForEach has been designed to make looping over dummies as fast as possible and so has some limitations.
Firstly, a single temporary Dummy object is created and on each function call it is updated with the current dummy data. This means that you should not try to store the Dummy object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new dummies inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all dummies are in
func	function	Function to call for each dummy
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the dummies in model m:

```
Dummy.ForEach(m, test);
function test(d)
{
// d is Dummy object
}
```

To call function test for all of the dummies in model m with optional object:

```
var data = { x:0, y:0 };
Dummy.ForEach(m, test, data);
function test(d, extra)
{
// d is Dummy object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Dummy objects for all of the dummies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get dummies from

Return type

Array of Dummy objects

Example

To make an array of Dummy objects for all of the dummies in model m

```
var d = Dummy.GetAll(m);
```

GetAssembly(index*[integer]*)

Description

Returns the information for an assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly you want the coordinates for. **Note that reference points start at 0, not 1**. 0 <= index < assemblies

Return type

Object with the following properties:

Name	Type	Description
label	integer	Assembly label
parent	integer	Parent assembly label
title	string	Assembly title

Example

To get the information for the 3rd assembly for dummy d:

```
var info = d.GetAssembly(2);
```

GetComments()

Description

Extracts the comments associated to a dummy.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the dummy d:

```
var comm_array = d.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Dummy objects for all of the flagged dummies in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get dummies from
flag	Flag	Flag set on the dummies that you want to retrieve

Return type

Array of Dummy objects

Example

To make an array of Dummy objects for all of the dummies in model m flagged with f

```
var d = Dummy.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Dummy object for a dummy ID.

Arguments

Name	Type	Description
Model	Model	Model to find the dummy in
number	integer	number of the dummy you want the Dummy object for

Return type

Dummy object (or null if dummy does not exist).

Example

To get the Dummy object for dummy 100 in model m

```
var d = Dummy.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Dummy property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Dummy.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	dummy property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Dummy property d.example is a parameter:

```
Options.property_parameter_names = true;
if (d.GetParameter(d.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Dummy property d.example is a parameter by using the GetParameter method:

```
if (d.ViewParameters().GetParameter(d.example) ) do_something...
```

GetPoint(index*[integer]*)

Description

Returns the information for a reference point

Arguments

Name	Type	Description
index	integer	The index of the reference point you want the information for. **Note that reference points start at 0, not 1**. 0 <= index < points

Return type

Object with the following properties:

Name	Type	Description
assembly	integer	Assembly label
csys	integer	Coordinate system
hpt	boolean	If point has been automatically created by PRIMER at the H-point
label	integer	Point label
node	integer	Node label (0 if coordinate)
rx	boolean	Point restrained rotationally in X
ry	boolean	Point restrained rotationally in Y
rz	boolean	Point restrained rotationally in Z
title	string	Point title
tx	boolean	Point restrained translationally in X
ty	boolean	Point restrained translationally in Y
tz	boolean	Point restrained translationally in Z
x	real	Node/point x coordinate
y	real	Node/point y coordinate
z	real	Node/point z coordinate

Example

To get the information for the 3rd reference point for dummy d:

```
var info = d.GetPoint(2);
```

GetPointData(rpt*[integer]*)

Description

Returns the coordinates of a reference point

Arguments

Name	Type	Description
rpt	integer	The reference point you want the coordinates for. **Note that reference points start at 0, not 1**.

Return type

Array containing the reference point coordinates

Example

To get the coordinates of the 3rd reference point for dummy d:

```
var c = d.GetPointData(2);
```

GetPointTitle(rpt*[integer]*)

Description

Returns the title of a reference point

Arguments

Name	Type	Description
rpt	integer	The reference point you want the title for. **Note that reference points start at 0, not 1**.

Return type

The reference point title

Example

To get the title of the 3rd reference point for dummy d:

```
var c = d.GetPointTitle(2);
```

Last(Model*[Model]*) [static]

Description

Returns the last dummy in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last dummy in

Return type

Dummy object (or null if there are no dummies in the model).

Example

To get the last dummy in model m:

```
var d = Dummy.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free dummy label in the model. Also see Dummy.FirstFreeLabel(), Dummy.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free dummy label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Dummy label.

Example

To get the last free dummy label in model m:

```
var label = Dummy.LastFreeLabel(m);
```

Next()

Description

Returns the next dummy in the model.

Arguments

No arguments

Return type

Dummy object (or null if there are no more dummies in the model).

Example

To get the dummy in model m after dummy d:

```
var d = d.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) dummy label in the model. Also see Dummy.FirstFreeLabel(), Dummy.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free dummy label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Dummy label.

Example

To get the next free dummy label in model m:

```
var label = Dummy.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a dummy.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only dummies from that model can be picked. If the argument is a Flag then only dummies that are flagged with *limit* can be selected. If omitted, or null, any dummies from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Dummy object (or null if not picked)

Example

To pick a dummy from model m giving the prompt 'Pick dummy from screen':

```
var d = Dummy.Pick('Pick dummy from screen', m);
```

Previous()

Description

Returns the previous dummy in the model.

Arguments

No arguments

Return type

Dummy object (or null if there are no more dummies in the model).

Example

To get the dummy in model m before dummy d:

```
var d = d.Previous();
```

RemovePoint(index*[integer]*)

Description

Removes a reference point from a dummy

Arguments

Name	Type	Description
index	integer	The index of the reference point you want to remove. **Note that reference points start at 0, not 1**. 0 <= index < points

Return type

no return value

Example

To remove for the 3rd reference point for dummy d:

```
d.RemovePoint(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the dummies in model m, from 1000000:

```
Dummy.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged dummies will be renumbered in
flag	Flag	Flag set on the dummies that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the dummies in model m flagged with f, from 1000000:

```
Dummy.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select dummies using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting dummies
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only dummies from that model can be selected. If the argument is a Flag then only dummies that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any dummies can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of dummies selected or null if menu cancelled

Example

To select dummies from model m, flagging those selected with flag f, giving the prompt 'Select dummies':

```
Dummy.Select(f, 'Select dummies', m);
```

To select dummies, flagging those selected with flag f but limiting selection to dummies flagged with flag l, giving the prompt 'Select dummies':

```
Dummy.Select(f, 'Select dummies', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the dummy.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the dummy

Return type

No return value

Example

To set flag f for dummy d:

```
d.SetFlag(f);
```

SetPoint(index*[integer]*, data*[object]*)

Description

Sets the data for a reference point in a dummy

Arguments

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>index</td><td>integer</td><td>The index of the reference point you want to set. Note that reference points start at 0, not 1. To add a new point use index points</td></tr>
<tr><td>data</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>assembly</td><td>integer</td><td>Assembly label</td></tr>
<tr><td>csys (optional)</td><td>integer</td><td>Coordinate system label</td></tr>
<tr><td>node (optional)</td><td>integer</td><td>Node label (not rewuired if using x, y and z)</td></tr>
<tr><td>rx (optional)</td><td>boolean</td><td>Point restrained rotationally in X</td></tr>
<tr><td>ry (optional)</td><td>boolean</td><td>Point restrained rotationally in Y</td></tr>
<tr><td>rz (optional)</td><td>boolean</td><td>Point restrained rotationally in Z</td></tr>
<tr><td>title (optional)</td><td>string</td><td>Title</td></tr>
<tr><td>tx (optional)</td><td>boolean</td><td>Point restrained translationally in X</td></tr>
<tr><td>ty (optional)</td><td>boolean</td><td>Point restrained translationally in Y</td></tr>
<tr><td>tz (optional)</td><td>boolean</td><td>Point restrained translationally in Z</td></tr>
<tr><td>x (optional)</td><td>real</td><td>X coordinate (not required if using node)</td></tr>
<tr><td>y (optional)</td><td>real</td><td>Y coordinate (not required if using node)</td></tr>
<tr><td>z (optional)</td><td>real</td><td>Z coordinate (not required if using node)</td></tr>
</table>
Object containing the reference point data. The properties can be: Object has the following properties:
</td></tr>
</table>

Return type

no return value

Example

To add a new reference point to dummy d assembly 5 at node 1000 with title "Example point" restrained in x:

```
var data = { assembly:5, node:1000, title:"Example point", tx:true };
d.SetPoint(d.points, data);
```

To add a new reference point to dummy d assembly 5 at (10, 20, 30) with title "Example point":

```
var data = { assembly:5, x:10, y:20, z:30, title:"Example point" };
d.SetPoint(d.points, data);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the dummy. The dummy will be sketched until you either call Dummy.Unsketch(), Dummy.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description

redraw (optional)	boolean	If model should be redrawn or not after the dummy is sketched. If omitted redraw is true. If you want to sketch several dummies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch dummy d:

```
d.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged dummies in the model. The dummies will be sketched until you either call Dummy.Unsketch(), Dummy.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged dummies will be sketched in
flag	Flag	Flag set on the dummies that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the dummies are sketched. If omitted redraw is true. If you want to sketch flagged dummies several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all dummies flagged with flag in model m:

```
Dummy.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing dummies should be counted. If false or omitted referenced but undefined dummies will also be included in the total.

Return type

number of dummies

Example

To get the total number of dummies in model m:

```
var total = Dummy.Total(m);
```

Unblank()

Description

Unblanks the dummy

Arguments

No arguments

Return type

No return value

Example

To unblank dummy d:

```
d.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the dummies in model m:

```
Dummy.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged dummies will be unblanked in
flag	Flag	Flag set on the dummies that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the dummies in model m flagged with f:

```
Dummy.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all dummies will be unset in
flag	Flag	Flag to unset on the dummies

Return type

No return value

Example

To unset the flag f on all the dummies in model m:

```
Dummy.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the dummy.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the dummy is unsketched. If omitted redraw is true. If you want to unsketch several dummies and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch dummy d:

```
d.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all dummies.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the dummies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all dummies in model m:

```
Dummy.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged dummies in the model.

Arguments

Name	Type	Description
Model	Model	Model that all dummies will be unsketched in
flag	Flag	Flag set on the dummies that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the dummies are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all dummies flagged with flag in model m:

```
Dummy.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with ’method chaining’ (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Dummy object.

Example

To check if Dummy property d.example is a parameter by using the Dummy.GetParameter() method:

```
if (d.ViewParameters().GetParameter(d.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for dummy. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for dummy d:

```
d.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this dummy.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for dummy d:

```
var xrefs = d.Xrefs();
```

File class

The File class allows you to read and write text files. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Copy(source*[string]*, dest*[string]*)
- Delete(filename*[string]*)
- DriveMapFilename(filename*[string]*, format*[constant]*)
- Exists(filename*[string]*)
- FindFiles(directory*[string]*, type (optional)*[constant]*)
- Get(url*[string]*, filename*[string]*, options (optional)*[object]*)
- IsAbsolute(filename*[string]*)
- IsDirectory(filename*[string]*)
- IsFile(filename*[string]*)
- IsReadable(filename*[string]*)
- IsWritable(filename*[string]*)
- Mkdir(directory*[string]*)
- Mktemp()
- Proxy(name*[string]*)
- ProxyPassword(name*[string]*)
- ProxyUsername(username*[string]*)
- ReadCSV(filename*[string]*, delimiter (optional)*[string]*, comment (optional)*[string]*)
- Rename(oldname*[string]*, newname*[string]*)
- Size(filename*[string]*)
- Upload(filename*[string]*, url*[string]*, options (optional)*[object]*)

Member functions

- Close()
- FindLineContaining(contain1*[string]*, contain2 (optional)*[string]*, contain3 (optional)*[string]*, ... containn (optional)*[string]*)
- FindLineStarting(start1*[string]*, start2 (optional)*[string]*, start3 (optional)*[string]*, ... startn (optional)*[string]*)
- Flush()
- ReadAll()
- ReadArrayBuffer(length (optional)*[integer]*)
- ReadChar()
- ReadLine()
- ReadLongLine()
- Seek(offset*[integer]*, origin (optional)*[constant]*)
- Tell()
- Write(string*[Any valid javascript type]*)
- WriteArrayBuffer(buffer*[ArrayBuffer]*, length (optional)*[integer]*)
- Writeln(string*[Any valid javascript type]*)

File constants

Name	Description
File.APPEND	Flag to open file for appending
File.BINARY	Flag to open file in binary mode. This will have no effect on unix/linux but for windows if a file is opened for writing with binary mode \n will not be translated to \r\n (CRLF), it will be written as \n (LF)
File.READ	Flag to open file for reading

File.UTF8	Flag to open file for reading as UTF-8 encoding.
File.WRITE	Flag to open file for writing

Constants for Find types

Name	Description
File.DIRECTORY	Find directories
File.FILE	Find files

Constants for Seek types

Name	Description
File.CURRENT	Seek relative to current file position
File.END	Seek relative to end of the file
File.START	Seek relative to start of the file

File properties

Name	Type	Description
filename (read only)	string	Name of the file
mode (read only)	constant	Mode the file was opened with (File.READ, File.WRITE etc)

Detailed Description

The File class gives you simple functions to read and write text files. The following simple example shows how to read from the file "/data/test/file.txt" and print each line read to the dialogue box:

```
var f, line;
f = new File("/data/test/file.txt", File.READ);
while ( (line = f.ReadLine()) != undefined)
{
    Message(line);
}
f.Close();
```

The following simple example shows how to write the numbers 1 to 10 to the file "/data/test/file.txt":

```
var n, line;
f = new File("/data/test/file.txt", File.WRITE);
for (n=1; n<=10; n++)
{
    f.Writeln(n);
}
f.Close();
```

See the documentation below for more details.

Constructor

new File(filename*[string]*, mode*[constant]*)

Description

Create a new File object for reading and writing text files.

Arguments

Name	Type	Description
filename	string	Filename of the file you want to read/write. If reading, the file must exist. If writing, the file will be overwritten (if it exists) if mode is File.WRITE, or if mode is File.APPEND it will be appended to if it exists, or created if it does not. When reading a file the filename can also be a URL (uniform resource locator) in which case the file will be read from the remote site. See File.Get() for more details on the format of the URL.
mode	constant	The mode to open the file with. Can be File.READ, File.WRITE or File.APPEND. For File.WRITE or File.APPEND it can also be ORed with File.BINARY if required. By default text is read and written as ASCII. To read/write text in utf-8 mode can also be ORed with File.UTF8 if required.

Return type

File object

Example

To create a new file object to read file "/data/test/file.txt"

```
var f = new File("/data/test/file.txt", File.READ);
```

Details of functions

Close()

Description

Close a file opened by a File object.

Arguments

No arguments

Return type

No return value

Example

To close File object f.

```
f.Close();
```

Copy(source*[string]*, dest*[string]*) [static]

Description

Copies a file

Arguments

Name	Type	Description
source	string	Source filename you want to copy.
dest	string	Destination filename you want to copy source file to.

Return type

true if copy successful, false otherwise.

Example

To copy the file "/data/test/file.key" to "/data/test/file.key_backup"

```
var copied = File.Copy("/data/test/file.key", "/data/test/file.key_backup");
```

Delete(filename*[string]*) [static]

Description

Deletes a file

Arguments

Name	Type	Description
filename	string	Filename you want to delete.

Return type

true if successful, false if not.

Example

To delete the file "/data/test/file.key"

```
var deleted = File.Delete("/data/test/file.key");
```

DriveMapFilename(filename*[string]*, format*[constant]*) [static]

Description

Changes a filename or directory name to the correct format for a specific operating system using the directory mappings (if present)

Arguments

Name	Type	Description
filename	string	Filename you want to drive map.
format	constant	The format for the file/directory name. Can be Include.NATIVE, Include.UNIX or Include.WINDOWS

Return type

string containing drive mapped filename

Example

If PRIMER has drive S: mapped to "/data" (by using the primer*drive_s, this*drive_s, d3plot*drive_s or oasys*drive_s preference)

```
var mapped = File.DriveMapFilename("/data/test/file.key", Include.WINDOWS);
```

mapped will be "S:\test\file.key".

```
var mapped = File.DriveMapFilename("S:\\test\\file.key", Include.UNIX);
```

mapped will be "/data/test/file.key".

Exists(filename*[string]*) [static]

Description

Check if a file exists. See also File.IsDirectory() and See also File.IsFile().

Arguments

Name	Type	Description
filename	string	Filename you want to check for existance.

Return type

true/false

Example

To see if the file "/data/test/file.key" exists

```
if (File.Exists("/data/test/file.key")) { do something }
```

FindFiles(directory*[string]*, type (optional)*[constant]*) [static]

Description

Find any files and/or directories in a directory.

Arguments

Name	Type	Description
directory	string	Directory to look for files/directories in.
type (optional)	constant	Type of things to find. Can be bitwise OR of File.FILE and File.DIRECTORY. If omitted only files will be returned.

Return type

Array of filenames/directories

Example

To return the filenames in the directory /data/test:

```
var fileList = File.FindFiles("/data/test")
```

To return the directories in the directory /data/test:

```
var fileList = File.FindFiles("/data/test", File.DIRECTORY)
```

To return the files and directories in the directory /data/test:

```
var fileList = File.FindFiles("/data/test", File.FILE|File.DIRECTORY)
```

FindLineContaining(contain1*[string]*, contain2 (optional)*[string]*, contain3 (optional)*[string]*, ... containn (optional)*[string]*)

Description

Reads a line from a file which contains **contain**, opened for reading by a File object. Although this is possible using core JavaScript functions this function should be significantly faster as most of the processing is done by Primer in C rather than in the JavaScript interpreter. To enable this function to be as fast as possible a maximum line length of 512 characters is used. If you expect a file to have lines longer than 512 characters then use ReadLongLine which allows lines of any length. If one argument is used then the line must contain that string. If more than one argument is used then lines which contain the string contain1 OR contain2 OR contain3 etc will be returned

Arguments

Name	Type	Description
contain1	string	String which matching lines must contain

contain2 (optional)	string	alternative string which matching lines must contain
contain3 (optional)	string	alternative string which matching lines must contain
... containn (optional)	string	alternative string which matching lines must contain

Return type

string read from file or `undefined` if end of file

Example

Loop, reading lines from File object f which contain 'example'.

```
var line;
while ( (line = f.FindLineContaining("example") ) != undefined)
{
}
```

FindLineStarting(start1*[string]*, start2 (optional)*[string]*, start3 (optional)*[string]*, ... startn (optional)*[string]*)

Description

Reads a line from a file which starts with start, opened for reading by a File object. Although this is possible using core JavaScript functions this function should be significantly faster as most of the processing is done by Primer in C rather than in the JavaScript interpreter. To enable this function to be as fast as possible a maximum line length of 512 characters is used. If you expect a file to have lines longer than 512 characters then use ReadLongLine which allows lines of any length. If one argument is used then the line must start with that string. If more than one argument is used then lines which start with start1 OR start2 OR start3 etc will be returned

Arguments

Name	Type	Description
start1	string	String which matching lines must start with
start2 (optional)	string	alternative string which matching lines must start with
start3 (optional)	string	alternative string which matching lines must start with
... startn (optional)	string	alternative string which matching lines must start with

Return type

string read from file or `undefined` if end of file

Example

Loop, reading lines from File object f which start 'example'.

```
var line;
while ( (line = f.FindLineStarting("example") ) != undefined)
{
}
```

Flush()

Description

Flushes a file opened for writing by a File object.

Arguments

No arguments

Return type

No return value

Example

To flush File object f.

```
f.Flush();
```

Get(url*[string]*, filename*[string]*, options (optional)*[object]*) [static]

Description

Get a file from a remote location. See also File.Proxy(), File.ProxyPassword() and File.ProxyUsername().

Arguments

Name	Type	Description
url	string	URL (uniform resource locator) of remote file you want to get. Currently http and ftp are supported. For http give the full address including the leading 'http://'. e.g. 'http://www.example.com/file.html'. For ftp an optional username and password can be given. e.g. 'ftp://ftp.example.com' retrieves the directory listing for the root directory. 'ftp://ftp.example.com/readme.txt' downloads the file readme.txt from the root directory. 'ftp://user:password@ftp.example.com/readme.txt' retrieves the readme.txt file from the user's home directory.
filename	string	Filename you want to save the file to.
options (optional)	object	Options for get. If 'username' and 'password' are set then basic authorization using the username and password will be used. Object has the following properties:

Name	Type	Description
password (optional)	string	Password
response (optional)	boolean	If set to true, then the response code will be returned instead of true/false. This can be used to retieve error messages and codes when the file is not returned successfully.
username (optional)	string	Username

Return type

true if file was successfully got, false otherwise.

Example

To get the file "http://www.example.com/file.html" and save it to C:\temp:

```
File.Get("http://www.example.com/file.html", "C:\temp\file.html");
```

IsAbsolute(filename*[string]*) [static]

Description

Check if a filename is absolute or relative.

Arguments

Name	Type	Description
filename	string	Filename you want to check.

Return type

true/false

Example

To see if the filename "/data/test" is absolute (which it is!)

```
if (File.IsAbsolute("/data/test")) { do something }
```

IsDirectory(filename*[string]*) [static]

Description

Check if a filename is a directory. See also File.Exists(), File.IsFile(), File.IsReadable() and File.IsWritable().

Arguments

Name	Type	Description
filename	string	Filename you want to check.

Return type

true/false

Example

To see if the filename "/data/test" is a directory

```
if (File.IsDirectory("/data/test")) { do something }
```

IsFile(filename*[string]*) [static]

Description

Check if a filename is a file. See also File.Exists(), File.IsDirectory(), File.IsReadable() and File.IsWritable().

Arguments

Name	Type	Description
filename	string	Filename you want to check.

Return type

true/false

Example

To see if the filename "/data/test" is a file

```
if (File.IsFile("/data/test")) { do something }
```

IsReadable(filename*[string]*) [static]

Description

Check if a filename has read permissions. See also File.Exists(), File.IsDirectory() and File.IsWritable().

Arguments

Name	Type	Description

filename	string	Filename you want to check.

Return type

true/false

Example

To see if the filename "/data/test" is readable

```
if (File.IsReadable("/data/test")) { do something }
```

IsWritable(filename*[string]*) [static]

Description

Check if a filename has write permissions. If *filename* exists and it is a file then it is checked to see if it can be opened with write (File.APPEND permissions). If *filename* exists and it is a directory then the directory is checked for write permission (can files be created in the directory). If *filename* does not exist then it is assumed to be a file and is checked to see if it can be opened for writing (File.WRITE permissions). See also File.Exists(), File.IsDirectory() and File.IsReadable().

Arguments

Name	Type	Description
filename	string	Filename you want to check.

Return type

true/false

Example

To see if the filename "/data/test" is writable

```
if (File.IsWritable("/data/test")) { do something }
```

Mkdir(directory*[string]*) [static]

Description

Make a directory. If Primer preference 'directory_permission' is set e.g.755 then this will apply (same as if set by chmod 755) ignoring any setting of umask. If there is no preference then the users current setting of umask will control permissions (same as system mkdir)

Arguments

Name	Type	Description
directory	string	The name of the directory you want to create.

Return type

true if successfully created, false if not.

Example

To make the directory "/data/test"

```
var success = File.Mkdir("/data/test");
```

Mktemp() [static]

Description

Make a temporary filename for writing a temporary file.

Arguments

No arguments

Return type

String name of temporary filename that can be used.

Example

To get a temp filename"

```
var filename = File.Mktemp();
```

Proxy(name*[string]*) [static]

Description

Set a proxy for files opened by http, ftp etc. See also File.Get(), File.ProxyPassword() and File.ProxyUsername().

Arguments

Name	Type	Description
name	string	The name of the proxy.

Return type

No return value

Example

To set the proxy to "http://example.proxy.com" using port 80:

```
File.Proxy("http://example.proxy.com:80");
```

ProxyPassword(name*[string]*) [static]

Description

Set a proxy password for files opened by http, ftp etc. See also File.Get(), File.Proxy() and File.ProxyUsername().

Arguments

Name	Type	Description
name	string	Password for the proxy server.

Return type

No return value

Example

To set the proxy password to "password":

```
File.ProxyPassword("password");
```

ProxyUsername(username*[string]*) [static]

Description

Set a proxy username for files opened by http, ftp etc. See also File.Get(), File.Proxy() and File.ProxyPassword().

Arguments

Name	Type	Description
username	string	The username for the proxy.

Return type

No return value

Example

To set the proxy username to "username":

```
File.ProxyUsername("username");
```

ReadAll()

Description

Reads **all** the remaining characters from a file opened for reading by a File object. As this function can read the entire file as a string be careful when reading large files as it will consume large amounts of memory.

Arguments

No arguments

Return type

String. Characters read from file or undefined if end of file

Example

Read all characters from File object f.

```
var c = f.ReadAll();
```

ReadArrayBuffer(length (optional)*[integer]*)

Description

Reads binary data from a file opened for reading by a File object. The data is returned as an ArrayBuffer object. For more details on how to use an ArrayBuffer see the following links:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Typed_arrays
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/ArrayBuffer
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypedArray
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/DataView.

Arguments

Name	Type	Description
length (optional)	integer	Number of bytes to try to read from the file. If omitted all the remaining data from the file will be read.

Return type

ArrayBuffer object or undefined if end of file

Example

To read data as 32bit unsigned integers from File object f.

```
var ab = f.ReadArrayBuffer();
var u32 = new Uint32Array(ab);
for (var i=0; i<u32.length; i++
{
    var value = u32[i];
}
```

ReadCSV(filename*[string]*, delimiter (optional)*[string]*, comment (optional)*[string]*) [static]

Description

Reads the input CSV file and returns an array of string arrays. If the CSV file has legitimate records the function returns an Array object containing sub-arrays of strings otherwise the function returns NULL. The lengths of all the sub-arrays are the same and equal to maximum number of fields in any of the records. For records in a CSV file having fewer fields, the respective sub-arrays are padded with NULL elements to the maximum array length.

Arguments

Name	Type	Description
filename	string	Filename you want to read CSV options from.
delimiter (optional)	string	Delimiter string to be used. Default is a comma (",").
comment (optional)	string	Comment string to be used. Default is a dollar sign ("$").

Return type

Array object containing string arrays.

Example

To Read CSV file "sample.csv" and print all records to a Window.

```
var csv_file_path = "C:\\sample.csv";
var records = "";
if(!File.Exists(csv_file_path))
{
    Window.Information("CSV file %s not present", csv_file_path);
    Exit();
}
var csv_array = File.ReadCSV(csv_file_path);
if(csv_array != null)
{
    for(var i = 0; i < csv_array.length; i++)
    {
        var record_array = csv_array[i];
        for(var j = 0; j < record_array.length; j++)
        {
            if(record_array[j] != null)
                records = records + record_array[j] + " , ";
        }
        records = records + "\n";
    }
}
Options.max_window_lines = csv_array.length;
Window.Information("File.ReadCSV Ouptut", records);
```

To Read CSV file "sample.csv" with delimiter string "::" and comment string "##".

```
var csv_array = File.ReadCSV(csv_file_path, "::", "##");
```

ReadChar()

Description

Reads a single character from a file opened for reading by a File object.

Arguments

No arguments

Return type

character read from file or

`undefined`

if end of file

Example

Loop, reading characters from File object f.

```
var c;
while ( (c = f.ReadChar()) != undefined) { ... }
```

ReadLine()

Description

Reads a line from a file opened for reading by a File object. To enable this function to be as fast as possible a maximum line length of 512 characters is used. If you expect a file to have lines longer than 512 characters then use ReadLongLine which allows lines of any length.

Arguments

No arguments

Return type

string read from file or

`undefined`

if end of file

Example

Loop, reading lines from File object f.

```
var line;
while ( (line = f.ReadLine()) != undefined) { ... }
```

ReadLongLine()

Description

Reads a line from a file opened for reading by a File object. The line can be any length. If your file has lines shorter than 512 characters then you may want to use ReadLine instead which is faster.

Arguments

No arguments

Return type

string read from file or

`undefined`

if end of file

Example

Loop, reading lines from File object f.

```
var line;
while ( (line = f.ReadLongLine()) != undefined) { ... }
```

Rename(oldname*[string]*, newname*[string]*) [static]

Description

Rename an existing file to have a different name.

Arguments

Name	Type	Description
oldname	string	Existing filename you want to rename

newname	string	New filename you want to rename to

Return type

true if successful, false if not.

Example

To rename the file "/data/test/file.key" to "/data/test/new_file.key"

```
var size = File.Rename("/data/test/file.key", "/data/test/new_file.key");
```

Seek(offset*[integer]*, origin (optional)*[constant]*)

Description

Set the current position for reading or writing in a File object.

Arguments

Name	Type	Description
offset	integer	Offset to seek to in the file
origin (optional)	constant	Origin for offset. Must be one of File.START, File.END or File.CURRENT. If omitted File.START will be used.

Return type

no return value

Example

To seek to the end of File f:

```
f.Seek(0, File.END);
```

To seek to the beginning of File f:

```
f.Seek(0, File.START);
```

To move forward 10 characters in File f:

```
f.Seek(10, File.CURRENT);
```

Size(filename*[string]*) [static]

Description

Return the size of a file in bytes

Arguments

Name	Type	Description
filename	string	Filename you want the size of.

Return type

size in bytes

Example

To get the size of the file "/data/test/file.key"

```
var size = File.Size("/data/test/file.key");
```

Tell()

Description

Return the current file position for a File object. Note that on Windows when reading files if the file is not opened with File.BINARY this may not return the correct file position for files with unix line endings.

Arguments

No arguments

Return type

integer

Example

To get the current file position for File f:

```
var pos = f.Tell();
```

Upload(filename*[string]*, url*[string]*, options (optional)*[object]*) [static]

Description

Uploads a file to a remote location. See also File.Proxy(), File.ProxyPassword() and File.ProxyUsername().

Arguments

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>filename</td><td>string</td><td>Filename you want to upload.</td></tr>
<tr><td>url</td><td>string</td><td>URL (uniform resource locator) of the remote location you want to upload the file to. Currently only http is supported. Give the full address including the leading ’http://’. e.g. ’http://www.example.com/file.html’.</td></tr>
<tr><td>options (optional)</td><td>object</td><td>Options for upload. If both of these are set then basic authorization using the username and password will be used. Object has the following properties:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>password (optional)</td><td>string</td><td>Password</td></tr>
<tr><td>username (optional)</td><td>string</td><td>Username</td></tr>
</table>
</td></tr>
</table>

Return type

true if file was successfully uploaded, false otherwise.

Example

To upload the file "C:\temp\file.txt" to "http://www.example.com/file.txt":

```
File.Upload("C:/temp/file.txt", "http://www.example.com/file.txt");
```

Write(string*[Any valid javascript type]*)

Description

Write a string to a file opened for writing by a File object. **Note that a carriage return is not added**.

Arguments

Name	Type	Description

string	Any valid javascript type	The string/item that you want to write

Return type

No return value

Example

To write string "Hello, world!" to File object f

```
f.Write("Hello, world!\n");
```

To write the title of model m to File object f

```
f.Write("The title of model 2 is " + m.title + "\n");
```

WriteArrayBuffer(buffer*[ArrayBuffer]*, length (optional)*[integer]*)

Description

Writes binary data to a file opened for writing by a File object. The data to write is an ArrayBuffer object. For more details on how to use an ArrayBuffer see the following links:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Typed_arrays
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/ArrayBuffer
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/TypedArray
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/DataView.

Arguments

Name	Type	Description
buffer	ArrayBuffer	ArrayBuffer to write to file
length (optional)	integer	Number of bytes to write to the file. If omitted all the data in the ArrayBuffer will be written (buffer.byteLength bytes)

Return type

No return value

Example

To write ArrayBuffer ab to File object f.

```
f.WriteArrayBuffer(ab);
```

Writeln(string*[Any valid javascript type]*)

Description

Write a string to a file opened for writing by a File object **adding a carriage return**.

Arguments

Name	Type	Description
string	Any valid javascript type	The string/item that you want to write

Return type

No return value

Example

To write string "Hello, world!" to File object f automatically adding a carriage return

```
f.Writeln("Hello, world!");
```

To write the title of model m to File object f automatically adding a carriage return

```
f.Writeln("The title of model 2 is " + m.title);
```

GeometrySurface class

The GeometrySurface class gives you access to surfaces in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- CalculateNormal(u*[real]*, y*[real]*)
- CalculatePoint(u*[real]*, v*[real]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetEdgeIndices()
- GetParameter(prop*[string]*)
- GetTriaIndices()
- GetVertices()
- Next()
- Previous()
- ProjectPoint(x*[real]*, y*[real]*, z*[real]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

GeometrySurface properties

Name	Type	Description
exists	logical	true if gsrf exists, false if referred to but not defined. (read only)
id	integer	GeometrySurface number. Also see the label property which is an alternative name for this. (read only)
include	integer	The Include file number that the gsrf is in.
label	integer	GeometrySurface number. Also see the id property which is an alternative name for this. (read only)
model	integer	The Model number that the surface is in.

Detailed Description

The GeometrySurface class allows you to create, modify, edit and manipulate surfaces cards. See the documentation below for more details.

Details of functions

AssociateComment(Comment[*Comment*])

Description

Associates a comment with a surface.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the surface

Return type

No return value

Example

To associate comment c to the surface s:

```
s.AssociateComment(c);
```

Blank()

Description

Blanks the surface

Arguments

No arguments

Return type

No return value

Example

To blank surface s:

```
s.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the surfaces in model m:

```
GeometrySurface.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged surfaces will be blanked in
flag	Flag	Flag set on the surfaces that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the surfaces in model m flagged with f:

```
GeometrySurface.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the surface is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if surface s is blanked:

```
if (s.Blanked() ) do_something...
```

CalculateNormal(u*[real]*, y*[real]*)

Description

Calculate the normal vector for a parametric point on a surface.

Arguments

Name	Type	Description
u	real	u parametric coordinate
y	real	v parametric coordinate

Return type

Array containing x, y and z values.

Example

To obtain the surface normal at parametric point (0.2, 0.3) on surface s:

```
var coords = s.CalculateNormal(0.2, 0.3);
```

CalculatePoint(u*[real]*, v*[real]*)

Description

Calculate the X, Y and Z coordinates for a parametric point on a surface.

Arguments

Name	Type	Description
u	real	u parametric coordinate
v	real	v parametric coordinate

Return type

Array containing x, y and z values.

Example

To obtain the coordinates of parametric point (0.2, 0.3) on surface s:

```
var coords = s.CalculatePoint(0.2, 0.3);
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the surface.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the surface

Return type

No return value

Example

To clear flag f for surface s:

```
s.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the surface.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

GeometrySurface object

Example

To copy surface s into surface z:

```
var z = s.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a surface.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the surface

Return type

No return value

Example

To detach comment c from the surface s:

```
s.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for surface. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for surface s:

```
s.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first surface in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first surface in

Return type

GeometrySurface object (or null if there are no surfaces in the model).

Example

To get the first surface in model m:

```
var s = GeometrySurface.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free surface label in the model. Also see GeometrySurface.LastFreeLabel(), GeometrySurface.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free surface label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

GeometrySurface label.

Example

To get the first free surface label in model m:

```
var label = GeometrySurface.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the surfaces in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be flagged in
flag	Flag	Flag to set on the surfaces

Return type

No return value

Example

To flag all of the surfaces with flag f in model m:

```
GeometrySurface.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the surface is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the surface

Return type

true if flagged, false if not.

Example

To check if surface s has flag f set on it:

```
if (s.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each surface in the model.
Note that ForEach has been designed to make looping over surfaces as fast as possible and so has some limitations.
Firstly, a single temporary GeometrySurface object is created and on each function call it is updated with the current surface data. This means that you should not try to store the GeometrySurface object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new surfaces inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces are in
func	function	Function to call for each surface

extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the surfaces in model m:

```
GeometrySurface.ForEach(m, test);
function test(s)
{
// s is GeometrySurface object
}
```

To call function test for all of the surfaces in model m with optional object:

```
var data = { x:0, y:0 };
GeometrySurface.ForEach(m, test, data);
function test(s, extra)
{
// s is GeometrySurface object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of GeometrySurface objects for all of the surfaces in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get surfaces from

Return type

Array of GeometrySurface objects

Example

To make an array of GeometrySurface objects for all of the surfaces in model m

```
var s = GeometrySurface.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a surface.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the surface s:

```
var comm_array = s.GetComments();
```

GetEdgeIndices()

Description

Return an array of all the edge indices for a surface (in pairs).

Arguments

No arguments

Return type

Array of indices

Example

To get edge indices for surface s

```
var edges = s.GetEdgeIndices();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of GeometrySurface objects for all of the flagged surfaces in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get surfaces from
flag	Flag	Flag set on the surfaces that you want to retrieve

Return type

Array of GeometrySurface objects

Example

To make an array of GeometrySurface objects for all of the surfaces in model m flagged with f

```
var s = GeometrySurface.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the GeometrySurface object for a surface ID.

Arguments

Name	Type	Description
Model	Model	Model to find the surface in
number	integer	number of the surface you want the GeometrySurface object for

Return type

GeometrySurface object (or null if surface does not exist).

Example

To get the GeometrySurface object for surface 100 in model m

```
var s = GeometrySurface.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a GeometrySurface property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the GeometrySurface.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	surface property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if GeometrySurface property s.example is a parameter:

```
Options.property_parameter_names = true;
if (s.GetParameter(s.example) ) do_something...
Options.property_parameter_names = false;
```

To check if GeometrySurface property s.example is a parameter by using the GetParameter method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

GetTriaIndices()

Description

Return an array of all the tria indices for a surface (in triplets).

Arguments

No arguments

Return type

Array of indices

Example

To get tria indices for surface s

```
var trias = s.GetTriaIndices();
```

GetVertices()

Description

Return an array of all the vertex coordinates for a surface (in triplets).

Arguments

No arguments

Return type

Array of indices

Example

To get vertex coordinates for surface s

```
var vertices = s.GetVertices();
```

Last(Model*[Model]*) [static]

Description

Returns the last surface in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last surface in

Return type

GeometrySurface object (or null if there are no surfaces in the model).

Example

To get the last surface in model m:

```
var s = GeometrySurface.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free surface label in the model. Also see GeometrySurface.FirstFreeLabel(), GeometrySurface.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free surface label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

GeometrySurface label.

Example

To get the last free surface label in model m:

```
var label = GeometrySurface.LastFreeLabel(m);
```

Next()

Description

Returns the next surface in the model.

Arguments

No arguments

Return type

GeometrySurface object (or null if there are no more surfaces in the model).

Example

To get the surface in model m after surface s:

```
var s = s.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) surface label in the model. Also see GeometrySurface.FirstFreeLabel(), GeometrySurface.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free surface label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

GeometrySurface label.

Example

To get the next free surface label in model m:

```
var label = GeometrySurface.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a surface.

Arguments

Name	Type	Description

prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only surfaces from that model can be picked. If the argument is a Flag then only surfaces that are flagged with *limit* can be selected. If omitted, or null, any surfaces from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

GeometrySurface object (or null if not picked)

Example

To pick a surface from model m giving the prompt 'Pick surface from screen':

```
var s = GeometrySurface.Pick('Pick surface from screen', m);
```

Previous()

Description

Returns the previous surface in the model.

Arguments

No arguments

Return type

GeometrySurface object (or null if there are no more surfaces in the model).

Example

To get the surface in model m before surface s:

```
var s = s.Previous();
```

ProjectPoint(x*[real]*, y*[real]*, z*[real]*)

Description

Project a point onto the surface.

Arguments

Name	Type	Description
x	real	X coordinate of point to project
y	real	Y coordinate of point to project
z	real	Z coordinate of point to project

Return type

Array containing u and v values.

Example

To obtain the projection of point (1, 2, 3) on to surface s:

```
var projection = s.ProjectPoint(1, 2, 3);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the surfaces in model m, from 1000000:

```
GeometrySurface.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged surfaces will be renumbered in
flag	Flag	Flag set on the surfaces that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the surfaces in model m flagged with f, from 1000000:

```
GeometrySurface.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select surfaces using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting surfaces
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only surfaces from that model can be selected. If the argument is a Flag then only surfaces that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any surfaces can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of surfaces selected or null if menu cancelled

Example

To select surfaces from model m, flagging those selected with flag f, giving the prompt 'Select surfaces':

```
GeometrySurface.Select(f, 'Select surfaces', m);
```

To select surfaces, flagging those selected with flag f but limiting selection to surfaces flagged with flag l, giving the prompt 'Select surfaces':

```
GeometrySurface.Select(f, 'Select surfaces', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the surface.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the surface

Return type

No return value

Example

To set flag f for surface s:

```
s.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the surface. The surface will be sketched until you either call GeometrySurface.Unsketch(), GeometrySurface.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the surface is sketched. If omitted redraw is true. If you want to sketch several surfaces and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch surface s:

```
s.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged surfaces in the model. The surfaces will be sketched until you either call GeometrySurface.Unsketch(), GeometrySurface.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged surfaces will be sketched in
flag	Flag	Flag set on the surfaces that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the surfaces are sketched. If omitted redraw is true. If you want to sketch flagged surfaces several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all surfaces flagged with flag in model m:

```
GeometrySurface.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing surfaces should be counted. If false or omitted referenced but undefined surfaces will also be included in the total.

Return type

number of surfaces

Example

To get the total number of surfaces in model m:

```
var total = GeometrySurface.Total(m);
```

Unblank()

Description

Unblanks the surface

Arguments

No arguments

Return type

No return value

Example

To unblank surface s:

```
s.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the surfaces in model m:

```
GeometrySurface.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged surfaces will be unblanked in
flag	Flag	Flag set on the surfaces that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the surfaces in model m flagged with f:

```
GeometrySurface.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all surfaces will be unset in
flag	Flag	Flag to unset on the surfaces

Return type

No return value

Example

To unset the flag f on all the surfaces in model m:

```
GeometrySurface.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the surface.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the surface is unsketched. If omitted redraw is true. If you want to unsketch several surfaces and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch surface s:

```
s.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all surfaces.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the surfaces are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all surfaces in model m:

```
GeometrySurface.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged surfaces in the model.

Arguments

Name	Type	Description
Model	Model	Model that all surfaces will be unsketched in
flag	Flag	Flag set on the surfaces that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the surfaces are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all surfaces flagged with flag in model m:

```
GeometrySurface.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

GeometrySurface object.

Example

To check if GeometrySurface property s.example is a parameter by using the GeometrySurface.GetParameter() method:

```
if (s.ViewParameters().GetParameter(s.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for surface. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for surface s:

```
s.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this surface.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for surface s:

```
var xrefs = s.Xrefs();
```

Graphics class

The Graphics class allows you to draw graphics in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- DepthTest(enable*[boolean]*)
- DrawingFunction(name*[function]*)
- FillColour(colour*[Colour]*)
- Finish()
- Line(x1*[real]*, y1*[real]*, z1*[real]*, x2*[real]*, y2*[real]*, z2*[real]*)
- LineColour(colour*[Colour]*)
- LineStyle(style*[constant]*)
- LineTo(x*[real]*, y*[real]*, z*[real]*)
- LineWidth(width*[Integer]*)
- MoveTo(x*[real]*, y*[real]*, z*[real]*)
- PolygonFinish()
- PolygonStart()
- Shape(shape*[constant]*, size*[integer]*)
- Start()
- Text(text*[String]*)
- TextColour(colour*[Colour]*)
- TextSize(size*[Integer]*)

Graphics constants

Name	Description
Graphics.CIRCLE	Circle shape. See Graphics.Shape() for use.
Graphics.DASHDOT_LINE	Dashed and dotted lines. See Graphics.LineStyle() for use.
Graphics.DASH_LINE	Dashed lines. See Graphics.LineStyle() for use.
Graphics.DIAMOND	Diamond shape. See Graphics.Shape() for use.
Graphics.DOT_LINE	Dotted lines. See Graphics.LineStyle() for use.
Graphics.FILLED_CIRCLE	Filled circle shape. See Graphics.Shape() for use.
Graphics.FILLED_DIAMOND	Filled diamond shape. See Graphics.Shape() for use.
Graphics.FILLED_HOURGLASS	Filled hourglass shape. See Graphics.Shape() for use.
Graphics.FILLED_SQUARE	Filled square shape. See Graphics.Shape() for use.
Graphics.HOURGLASS	Hourglass shape. See Graphics.Shape() for use.
Graphics.POINT	Point shape. See Graphics.Shape() for use.
Graphics.SOLID_LINE	Solid lines. See Graphics.LineStyle() for use.
Graphics.SQUARE	Square shape. See Graphics.Shape() for use.

Detailed Description

The Graphics class gives you access to functions to draw lines, shapes etc on the graphics screen in PRIMER. For example the following will draw a solid thick red line on the screen:

```
Graphics.Start();
Graphics.LineWidth(3);
Graphics.LineColour(Colour.RED);
Graphics.LineStyle(Graphics.SOLID_LINE);
Graphics.Line(0, 0, 0, 10, 20, 30);
Graphics.Finish();
```

The drawing commands must be between

```
Graphics.Start()
```

and

```
Graphics.Finish()
```

or else nothing will be seen. This is suitable for sketching but the line will disappear if the graphics are redrawn or any dynamic viewing is done. To draw graphics which will stay on the screen even if dynamic viewing or a redraw is done you have to register a function using Graphics.DrawingFunction() which will be called every time the graphics are redrawn by PRIMER. e.g:

```
var w = new Window("Graphics test", 0.8, 1.0, 0.5, 0.6);
var e = new Widget(w, Widget.BUTTON,  1, 21, 1, 7, "Exit");
e.onClick  = Exit;
do_draw();
Graphics.DrawingFunction(do_draw);
w.Show();
////////////////////////////////////////////////////////////////////////////
function do_draw()
{
    Graphics.Start();
    Graphics.LineWidth(3);
    Graphics.LineColour(Colour.RED);
    Graphics.LineStyle(Graphics.SOLID_LINE);
    Graphics.Line(0, 0, 0, 10, 20, 30);
    Graphics.Finish();
}
```

See the documentation below for more details.

Details of functions

DepthTest(enable*[boolean]*) [static]

Description

Allows depth testing (hidden surface removal) to be turned on or off. Temporarily turning depth testing off may be used to ensure that an item (e.g. some text) is always drawn in front and will not be obscured.

Arguments

Name	Type	Description
enable	boolean	Whether depth testing (hidden surface removal) is performed (true) or not (false)

Return type

No return value

Example

To turn off depth testing:

```
Graphics.DepthTest(false);
```

To turn depth testing back on:

```
Graphics.DepthTest(true);
```

DrawingFunction(name*[function]*) [static]

Description

Set the function to draw graphics from javaScript. This function will be called each time the graphics are redrawn after PRIMER has finished drawing everything else. This allows you to add extra items to the graphics.
To remove the graphics drawing function use Graphics.DrawingFunction(null).
It is the responsibility of the script developer to ensure that any objects or variables that are used in the drawing function do not refer to items in Primer that no longer exist. Not doing so may cause PRIMER to crash. For example, if you use some Node objects in the drawing function that refer to nodes in model 1 and you delete the model, when the graphics are redrawn PRIMER may crash as the nodes referred to by the Node objects no longer exist. You should either remove the drawing function by calling Graphics.DrawingFunction(null) or set the Node variables to null (and test that they exist before using them) in your drawing function **before** deleting the model.

Arguments

Name	Type	Description
name	function	The name of the function (or null to remove a function)

Return type

No return value

Example

To set function MyRedrawFunction as the Graphics drawing function

```
Graphics.DrawingFunction(MyRedrawFunction);
```

FillColour(colour*[Colour]*) [static]

Description

Sets the colour for drawing polygons. See the Colour class for more details on colours.

Arguments

Name	Type	Description
colour	Colour	The colour you want to fill polygons with

Return type

No return value

Example

To Set the current fill colour to red:

```
Graphics.FillColour(Colour.RED);
```

or

```
Graphics.FillColour( Colour.RGB(255, 0, 0) );
```

Finish() [static]

Description

Finish any graphics. See also Graphics.Start(). This **must** be used to finish drawing.

Arguments

No arguments

Return type

No return value

Example

To finish any graphics operations:

```
Graphics.Finish();
```

Line(x1*[real]*, y1*[real]*, z1*[real]*, x2*[real]*, y2*[real]*, z2*[real]*) [static]

Description

Draws a line from (x1, y1, z1) to (x2, y2, z2). See also Graphics.LineTo() and Graphics.MoveTo()

Arguments

Name	Type	Description
x1	real	X coordinate of point 1
y1	real	Y coordinate of point 1
z1	real	Z coordinate of point 1
x2	real	X coordinate of point 2
y2	real	Y coordinate of point 2
z2	real	Z coordinate of point 2

Return type

No return value

Example

To draw a line from (0.0, 0.0, 0.0) to (10.0, 20.0, 30.0)

```
Graphics.Line(0.0, 0.0, 0.0, 10.0, 20.0, 30.0);
```

LineColour(colour*[Colour]*) [static]

Description

Sets the colour for drawing lines. See the Colour class for more details on colours.

Arguments

Name	Type	Description
colour	Colour	The colour you want to draw lines with

Return type

No return value

Example

To Set the current drawing colour to red:

```
Graphics.LineColour(Colour.RED);
```

or

```
Graphics.LineColour( Colour.RGB(255, 0, 0) );
```

LineStyle(style*[constant]*) [static]

Description

Sets the style for drawing lines.

Arguments

Name	Type	Description
style	constant	The style to draw lines with. Can be: Graphics.SOLID_LINE, Graphics.DASH_LINE, Graphics.DASHDOT_LINE or Graphics.DOT_LINE

Return type

No return value

Example

To Set the current line style to 3:

```
Graphics.LineStyle(3);
```

LineTo(x*[real]*, y*[real]*, z*[real]*) [static]

Description

Draws a line from the current point to (x, y, z). After drawing the line the current point will be (x, y, z). See also Graphics.Line() and Graphics.MoveTo()

Arguments

Name	Type	Description
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate

Return type

No return value

Example

To draw a line from the current point to (10.0, 20.0, 30.0):

```
Graphics.LineTo(10.0, 20.0, 30.0);
```

LineWidth(width*[Integer]*) [static]

Description

Sets the width for drawing lines.

Arguments

Name	Type	Description
width	Integer	The width to draw lines with

Return type

No return value

Example

To Set the current line width to 3:

```
Graphics.LineWidth(3);
```

MoveTo(x*[real]*, y*[real]*, z*[real]*) [static]

Description

Sets the current point to (x, y, z). See also Graphics.Line() and Graphics.LineTo()

Arguments

Name	Type	Description
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate

Return type

No return value

Example

To set the current point to (10.0, 20.0, 30.0):

```
Graphics.MoveTo(10.0, 20.0, 30.0);
```

PolygonFinish() [static]

Description

Ends drawing a polygon. See also Graphics.PolygonStart()

Arguments

No arguments

Return type

No return value

Example

To draw a red square:

```
Graphics.FillColour(Colour.RED);
Graphics.MoveTo(0.0, 0.0, 0.0);
Graphics.PolygonStart();
Graphics.MoveTo(10.0, 0.0, 0.0);
Graphics.MoveTo(10.0, 10.0, 0.0);
Graphics.MoveTo(0.0, 10.0, 0.0);
Graphics.PolygonFinish();
```

PolygonStart() [static]

Description

Starts drawing a polygon. See also Graphics.PolygonFinish()

Arguments

No arguments

Return type

No return value

Example

To draw a red square:

```
Graphics.FillColour(Colour.RED);
Graphics.MoveTo(0.0, 0.0, 0.0);
Graphics.PolygonStart();
Graphics.MoveTo(10.0, 0.0, 0.0);
Graphics.MoveTo(10.0, 10.0, 0.0);
Graphics.MoveTo(0.0, 10.0, 0.0);
Graphics.PolygonFinish();
```

Shape(shape*[constant]*, size*[integer]*) [static]

Description

Draws a simple shape.

Arguments

Name	Type	Description
shape	constant	The style to draw lines with. Can be: Graphics.POINT, Graphics.SQUARE, Graphics.CIRCLE, Graphics.DIAMOND, Graphics.HOURGLASS, Graphics.FILLED_SQUARE, Graphics.FILLED_CIRCLE, Graphics.FILLED_DIAMOND or Graphics.FILLED_HOURGLASS
size	integer	Size the shape should be drawn at.

Return type

No return value

Example

To draw a filled square at (10, 20, 30) at size 10:

```
Graphics.MoveTo(10, 20, 30);
Graphics.Shape(Graphics.FILLED_SQUARE, 10);
```

Start() [static]

Description

Start any graphics. See also Graphics.Finish(). This **must** be used before any drawing is done.

Arguments

No arguments

Return type

No return value

Example

To start drawing graphics:

```
Graphics.Start();
```

Text(text*[String]*) [static]

Description

Draws text at current position. See Graphics.MoveTo() to set the current position.

Arguments

Name	Type	Description
text	String	The text to write

Return type

No return value

Example

To write "Example" at (10, 20, 30):

```
Graphics.MoveTo(10, 20, 30);
Graphics.Text("Example");
```

TextColour(colour*[Colour]*) [static]

Description

Sets the colour for drawing text. See the Colour class for more details on colours.

Arguments

Name	Type	Description
colour	Colour	The colour you want to draw text with

Return type

No return value

Example

To Set the current text drawing colour to red:

```
Graphics.TextColour(Colour.RED);
```

or

```
Graphics.TextColour( Colour.RGB(255, 0, 0) );
```

TextSize(size*[Integer]*) [static]

Description

Sets the size for drawing text.

Arguments

Name	Type	Description
size	Integer	The size to draw text with

Return type

No return value

Example

To Set the current text size to 30:

```
Graphics.TextSize(30);
```

Group class

The Group class gives you access to groups in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetDataAll(type*[string]*, index*[integer]*)
- GetDataList(type*[string]*, index*[integer]*)
- GetDataRange(type*[string]*, index*[integer]*)
- GetParameter(prop*[string]*)
- GetTotalAll(type*[string]*)
- GetTotalList(type*[string]*)
- GetTotalRange(type*[string]*)
- GetTotals(type*[string]*)
- GetType(row*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()

- RemoveDataAll(type*[string]*, index*[integer]*)
- RemoveDataList(type*[string]*, index*[integer]*)
- RemoveDataRange(type*[string]*, index*[integer]*)
- SetDataAll(type*[string]*, index*[integer]*, data*[Array of data]*)
- SetDataList(type*[string]*, index*[integer]*, data*[Array of data]*)
- SetDataRange(type*[string]*, index*[integer]*, data*[Array of data]*)
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

Group constants

Name	Description
Group.ADD	Add contents to group
Group.REMOVE	Remove contents from group

Group properties

Name	Type	Description
exists	logical	true if group exists, false if referred to but not defined (read only)
include	integer	The Include file number that the group is in
label	integer	Group number
lock	logical	Whether Group contents are locked against deletion.
model	integer	The Model number that the group is in.
numtypes (read only)	integer	Number of types in the group.
title	string	Group title

Detailed Description

The Group class allows you to create, modify, edit and manipulate groups. See the documentation below for more details.

Constructor

new Group(Model*[Model]*, label*[integer]*, title (optional)*[string]*)

Description

Create a new Group object.

Arguments

Name	Type	Description
Model	Model	Model that Group will be created in
label	integer	Group number.
title (optional)	string	Title for the group

Return type

Group object

Example

To create a new group 99 in model m with title "Example":

```
var g = new Group(m, 99, "Example");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a group.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the group

Return type

No return value

Example

To associate comment c to the group g:

```
g.AssociateComment(c);
```

Blank()

Description

Blanks the group

Arguments

No arguments

Return type

No return value

Example

To blank group g:

```
g.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the groups in the model.

Arguments

Name	Type	Description

Model	Model	Model that all groups will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the groups in model m:

```
Group.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged groups will be blanked in
flag	Flag	Flag set on the groups that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the groups in model m flagged with f:

```
Group.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the group is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if group g is blanked:

```
if (g.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse group g:

```
g.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the group.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the group

Return type

No return value

Example

To clear flag f for group g:

```
g.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the group.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Group object

Example

To copy group g into group z:

```
var z = g.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a group.

Arguments

Name	Type	Description
Model	Model	Model that the group will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

Group object (or null if not made)

Example

To start creating a group g in model m:

```
var g = Group.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a group.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the group

Return type

No return value

Example

To detach comment c from the group g:

```
g.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit group g:

```
g.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for group. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for group g:

```
g.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first group in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first group in

Return type

Group object (or null if there are no groups in the model).

Example

To get the first group in model m:

```
var g = Group.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free group label in the model. Also see Group.LastFreeLabel(), Group.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free group label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Group label.

Example

To get the first free group label in model m:

```
var label = Group.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the groups in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all groups will be flagged in
flag	Flag	Flag to set on the groups

Return type

No return value

Example

To flag all of the groups with flag f in model m:

```
Group.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the group is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the group

Return type

true if flagged, false if not.

Example

To check if group g has flag f set on it:

```
if (g.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each group in the model.
Note that ForEach has been designed to make looping over groups as fast as possible and so has some limitations. Firstly, a single temporary Group object is created and on each function call it is updated with the current group data. This means that you should not try to store the Group object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new groups inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all groups are in
func	function	Function to call for each group
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the groups in model m:

```
Group.ForEach(m, test);
function test(g)
{
// g is Group object
}
```

To call function test for all of the groups in model m with optional object:

```
var data = { x:0, y:0 };
Group.ForEach(m, test, data);
function test(g, extra)
{
// g is Group object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Group objects for all of the groups in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get groups from

Return type

Array of Group objects

Example

To make an array of Group objects for all of the groups in model m

```
var g = Group.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a group.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the group g:

```
var comm_array = g.GetComments();
```

GetDataAll(type*[string]*, index*[integer]*)

Description

Returns 'all' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'all' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index < Group.GetTotalAll()

Return type

An array containing data [Group.ADD or Group.REMOVE, BOX (if defined)].

Example

To get the data for the 3rd SHELL 'all' row in group g:

```
var data = g.GetDataAll("SHELL", 2);
```

GetDataList(type*[string]*, index*[integer]*)

Description

Returns 'list' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'list' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index < Group.GetTotalList()

Return type

An array containing data [Group.ADD or Group.REMOVE, ITEM1 (if defined), ITEM2 (if defined), ITEM3 (if defined), ITEM4 (if defined), ITEM5 (if defined), BOX (if defined)].

Example

To get the data for the 3rd SHELL 'list' row in group g:

```
var data = g.GetDataList("SHELL", 2);
```

GetDataRange(type*[string]*, index*[integer]*)

Description

Returns 'range' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'range' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index < Group.GetTotalRange()

Return type

An array containing data [Group.ADD or Group.REMOVE, START, END, BOX (if defined)].

Example

To get the data for the 3rd SHELL 'range' row in group g:

```
var data = g.GetDataRange("SHELL", 2);
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Group objects for all of the flagged groups in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get groups from
flag	Flag	Flag set on the groups that you want to retrieve

Return type

Array of Group objects

Example

To make an array of Group objects for all of the groups in model m flagged with f

```
var g = Group.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Group object for a group ID.

Arguments

Name	Type	Description

Model	Model	Model to find the group in
number	integer	number of the group you want the Group object for

Return type

Group object (or null if group does not exist).

Example

To get the Group object for group 100 in model m

```
var g = Group.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Group property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Group.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	group property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Group property g.example is a parameter:

```
Options.property_parameter_names = true;
if (g.GetParameter(g.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Group property g.example is a parameter by using the GetParameter method:

```
if (g.ViewParameters().GetParameter(g.example) ) do_something...
```

GetTotalAll(type*[string]*)

Description

Returns the total number of 'all' rows for a type in a group

Arguments

Name	Type	Description
type	string	The type of the item

Return type

The number of 'all' rows defined

Example

To get the total number of shell 'all' rows in group g:

```
var nrow = g.GetTotalAll("SHELL");
```

GetTotalList(type*[string]*)

Description

Returns the total number of 'list' rows for a type in a group

Arguments

Name	Type	Description
type	string	The type of the item

Return type

The number of 'list' rows defined

Example

To get the total number of shell 'list' rows in group g:

```
var nrow = g.GetTotalList("SHELL");
```

GetTotalRange(type*[string]*)

Description

Returns the total number of 'range' rows for a type in a group

Arguments

Name	Type	Description
type	string	The type of the item

Return type

The number of 'range' rows defined

Example

To get the total number of shell 'range' rows in group g:

```
var nrow = g.GetTotalRange("SHELL");
```

GetTotals(type*[string]*)

Description

Returns the total number of 'all', 'list' and 'range' rows for a type in a group

Arguments

Name	Type	Description
type	string	The type of the item

Return type

Array containing number of ’all’, ’list’ and ’range’ rows defined or null if type not in group.

Example

To get the total number of shell ’all’, ’list’ and ’range’ rows in group g:

```
var totals = g.GetTotals("SHELL");
var nall   = totals[0];
var nlist  = totals[1];
var nrange = totals[2];
```

GetType(row*[integer]*)

Description

Returns the type for an entry in a group

Arguments

Name	Type	Description
row	integer	The entry in the group types that you want the type for. **Note that entries start at 0, not 1**

Return type

The type of the item (string)

Example

To list the types that are present in group g:

```
for (var t=0; t<g.numtypes; t++)
{
    var type = g.GetType(t);
    Message(type);
}
```

Keyword()

Description

Returns the keyword for this group. **Note that a carriage return is not added**. See also Group.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for goup g:

```
var key = g.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the Group. **Note that a carriage return is not added**. See also Group.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for Group g:

```
var cards = g.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last group in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last group in

Return type

Group object (or null if there are no groups in the model).

Example

To get the last group in model m:

```
var g = Group.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free group label in the model. Also see Group.FirstFreeLabel(), Group.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free group label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Group label.

Example

To get the last free group label in model m:

```
var label = Group.LastFreeLabel(m);
```

Next()

Description

Returns the next group in the model.

Arguments

No arguments

Return type

Group object (or null if there are no more groups in the model).

Example

To get the group in model m after group g:

```
var g = g.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) group label in the model. Also see Group.FirstFreeLabel(), Group.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free group label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Group label.

Example

To get the next free group label in model m:

```
var label = Group.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a group.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only groups from that model can be picked. If the argument is a Flag then only groups that are flagged with *limit* can be selected. If omitted, or null, any groups from any model can be selected. from any model.

modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Group object (or null if not picked)

Example

To pick a group from model m giving the prompt 'Pick group from screen':

```
var g = Group.Pick('Pick group from screen', m);
```

Previous()

Description

Returns the previous group in the model.

Arguments

No arguments

Return type

Group object (or null if there are no more groups in the model).

Example

To get the group in model m before group g:

```
var g = g.Previous();
```

RemoveDataAll(type*[string]*, index*[integer]*)

Description

Removes 'all' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'all' row you want to Remove. **Note that indices start at 0, not 1**. 0 <= index < Group.GetTotalAll()

Return type

No return value

Example

To remove the data for the 3rd SHELL 'all' row in group g:

```
g.RemoveDataAll("SHELL", 2);
```

RemoveDataList(type*[string]*, index*[integer]*)

Description

Removes 'list' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'list' row you want to Remove. **Note that indices start at 0, not 1.** 0 <= index < Group.GetTotalList()

Return type

No return value

Example

To remove the data for the 3rd SHELL 'list' row in group g:

```
g.RemoveDataList("SHELL", 2);
```

RemoveDataRange(type*[string]*, index*[integer]*)

Description

Removes 'range' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'range' row you want to Remove. **Note that indices start at 0, not 1.** 0 <= index < Group.GetTotalRange()

Return type

No return value

Example

To remove the data for the 3rd SHELL 'range' row in group g:

```
g.RemoveDataRange("SHELL", 2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all groups will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the groups in model m, from 1000000:

```
Group.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged groups will be renumbered in
flag	Flag	Flag set on the groups that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the groups in model m flagged with f, from 1000000:

```
Group.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select groups using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting groups
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only groups from that model can be selected. If the argument is a Flag then only groups that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any groups can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of groups selected or null if menu cancelled

Example

To select groups from model m, flagging those selected with flag f, giving the prompt 'Select groups':

```
Group.Select(f, 'Select groups', m);
```

To select groups, flagging those selected with flag f but limiting selection to groups flagged with flag l, giving the prompt 'Select groups':

```
Group.Select(f, 'Select groups', l);
```

SetDataAll(type*[string]*, index*[integer]*, data*[Array of data]*)

Description

Sets 'all' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'all' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index <= Group.GetTotalAll()
data	Array of data	An array containing data [Group.ADD or Group.REMOVE, BOX (if defined)].

Return type

No return value

Example

To set the data for the 3rd SHELL 'all' row in group g to 'add box 10':

```
var data = [Group.ADD, 10];
g.SetDataAll("SHELL", 2, data);
```

SetDataList(type*[string]*, index*[integer]*, data*[Array of data]*)

Description

Sets 'list' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'list' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index <= Group.GetTotalList()
data	Array of data	An array containing data [Group.ADD or Group.REMOVE, ITEM1 (if defined), ITEM2 (if defined), ITEM3 (if defined), ITEM4 (if defined), ITEM5 (if defined), BOX (if defined)].

Return type

No return value

Example

To set the data for the 3rd SHELL 'list' row in group g to 'add 1 2 box 10':

```
var data = [Group.ADD, 1, 2, 0, 0, 0, 10];
g.SetDataList("SHELL", 2, data);
```

SetDataRange(type*[string]*, index*[integer]*, data*[Array of data]*)

Description

Sets 'range' data for a given row number and type in the group.

Arguments

Name	Type	Description
type	string	The type of the item
index	integer	Index of 'all' row you want the data for. **Note that indices start at 0, not 1**. 0 <= index <= Group.GetTotalRange()
data	Array of data	An array containing data [Group.ADD or Group.REMOVE, START, END, BOX (if defined)].

Return type

No return value

Example

To set the data for the 3rd SHELL 'range' row in group g to 'add 100 200 box 10':

```
var data = [Group.ADD, 100, 200, 10];
g.SetDataRange("SHELL", 2, data);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the group.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the group

Return type

No return value

Example

To set flag f for group g:

```
g.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the group. The group will be sketched until you either call Group.Unsketch(), Group.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the group is sketched. If omitted redraw is true. If you want to sketch several groups and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch group g:

```
g.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged groups in the model. The groups will be sketched until you either call Group.Unsketch(), Group.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged groups will be sketched in
flag	Flag	Flag set on the groups that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the groups are sketched. If omitted redraw is true. If you want to sketch flagged groups several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all groups flagged with flag in model m:

```
Group.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of groups in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing groups should be counted. If false or omitted referenced but undefined groups will also be included in the total.

Return type

number of groups

Example

To get the total number of groups in model m:

```
var total = Group.Total(m);
```

Unblank()

Description

Unblanks the group

Arguments

No arguments

Return type

No return value

Example

To unblank group g:

```
g.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all groups will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the groups in model m:

```
Group.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged groups will be unblanked in
flag	Flag	Flag set on the groups that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the groups in model m flagged with f:

```
Group.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all groups will be unset in
flag	Flag	Flag to unset on the groups

Return type

No return value

Example

To unset the flag f on all the groups in model m:

```
Group.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the group.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the group is unsketched. If omitted redraw is true. If you want to unsketch several groups and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch group g:

```
g.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all groups.

Arguments

Name	Type	Description
Model	Model	Model that all groups will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the groups are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all groups in model m:

```
Group.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged groups in the model.

Arguments

Name	Type	Description
Model	Model	Model that all groups will be unsketched in
flag	Flag	Flag set on the groups that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the groups are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all groups flagged with flag in model m:

```
Group.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Group object.

Example

To check if Group property g.example is a parameter by using the Group.GetParameter() method:

```
if (g.ViewParameters().GetParameter(g.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for group. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for group g:

```
g.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this group.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for group g:

```
var xrefs = g.Xrefs();
```

---

toString()

Description

Creates a string containing the Group data in keyword format. Note that this contains the keyword header and the keyword cards. See also Group.Keyword() and Group.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for Group g in keyword format

```
var s = g.toString();
```

Image class

The Image class enables writing bitmaps to file. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- WriteBMP(filename*[string]*, resolution (optional)*[constant]*, 8bit (optional)*[boolean]*, options (optional)*[constant]*)
- WriteGIF(filename*[string]*, resolution (optional)*[constant]*, palette (optional)*[constant]*)
- WriteJPEG(filename*[string]*, resolution (optional)*[constant]*, quality (optional)*[integer]*)
- WritePNG(filename*[string]*, resolution (optional)*[constant]*, 8bit (optional)*[boolean]*, palette (optional)*[constant]*)

Image constants

Name	Description
Image.COMPRESS	If compression is done for 8 bit bmp images.
Image.DITHER	If dithering is done for 8 bit images.
Image.OPTIMISE	If palette optimisation is done for 8 bit images.
Image.SCREEN	Image will be created at screen resolution.
Image.X2	Image will be created at 2x screen resolution.
Image.X4	Image will be created at 4x screen resolution.

Detailed Description

The Image class enables you to write BMP, GIF, JPEG or PNG images from PRIMER. See the documentation below for more details.

Details of functions

WriteBMP(filename*[string]*, resolution (optional)*[constant]*, 8bit (optional)*[boolean]*, options (optional)*[constant]*) [static]

Description

Create a bmp image of the current screen image

Arguments

Name	Type	Description
filename	string	Filename you want to write. The file will be overwritten if it already exists.
resolution (optional)	constant	The resolution to write the image at. Can be Image.SCREEN, Image.X2 or Image.X4. If omitted screen resolution will be used

8bit (optional)	boolean	BMP images can be written using either 8 bit (256 colours) or 24 bit (16 million colours). If this is true then an 8 bit image will be written. If false (or omitted) a 24 bit image will be written.
options (optional)	constant	For 8 bit images (see '8bit' argument) the palette can be optimised (Image.OPTIMISE) and/or dithered (Image.DITHER) and/or compressed (Image.COMPRESS) If 0 (or omitted) no palette optimising, dithering or compression will be done.

Return type

No return value

Example

To create a 24 bit png file "/data/test/image.png" at 2x screen resolution

```
Image.WriteBMP("/data/test/image.bmp", Image.X2);
```

WriteGIF(filename*[string]*, resolution (optional)*[constant]*, palette (optional)*[constant]*) [static]

Description

Create a gif image of the current screen image

Arguments

Name	Type	Description
filename	string	Filename you want to write. The file will be overwritten if it already exists.
resolution (optional)	constant	The resolution to write the image at. Can be Image.SCREEN, Image.X2 or Image.X4. If omitted screen resolution will be used
palette (optional)	constant	The palette can be optimised (Image.OPTIMISE) and/or dithered (Image.DITHER). If 0 (or omitted) no palette optimising or dithering will be done.

Return type

No return value

Example

To create a gif file "/data/test/image.gif" at 2x screen resolution

```
Image.WriteGIF("/data/test/image.gif", Image.X2);
```

WriteJPEG(filename*[string]*, resolution (optional)*[constant]*, quality (optional)*[integer]*) [static]

Description

Create a jpeg image of the current screen image

Arguments

Name	Type	Description
filename	string	Filename you want to write. The file will be overwritten if it already exists.
resolution (optional)	constant	The resolution to write the image at. Can be Image.SCREEN, Image.X2 or Image.X4. If omitted screen resolution will be used
quality (optional)	integer	Quality of the image in percent. Can be in the range [10,100]. If omitted, the quality is 90.

Return type

No return value

Example

To create a jpeg file "/data/test/image.jpg" at 2x screen resolution

```
Image.WriteJPEG("/data/test/image.jpg", Image.X2);
```

WritePNG(filename*[string]*, resolution (optional)*[constant]*, 8bit (optional)*[boolean]*, palette (optional)*[constant]*) [static]

Description

Create a png image of the current screen image

Arguments

Name	Type	Description
filename	string	Filename you want to write. The file will be overwritten if it already exists.
resolution (optional)	constant	The resolution to write the image at. Can be Image.SCREEN, Image.X2 or Image.X4. If omitted screen resolution will be used
8bit (optional)	boolean	PNG images can be written using either 8 bit (256 colours) or 24 bit (16 million colours). If this is true then an 8 bit image will be written. If false (or omitted) a 24 bit image will be written.
palette (optional)	constant	For 8 bit images (see ’8bit’ argument) the palette can be optimised (Image.OPTIMISE) and/or dithered (Image.DITHER). If 0 (or omitted) no palette optimising or dithering will be done.

Return type

No return value

Example

To create a 24 bit png file "/data/test/image.png" at 2x screen resolution

```
Image.WritePNG("/data/test/image.png", Image.X2);
```

Mechanism class

The Mechanism class gives you access to mechanism cards in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AddNodeSetToAssembly(index*[integer]*, nsid*[integer]*)
- AddPartSetToAssembly(index*[integer]*, psid*[integer]*)
- AddPartToAssembly(index*[integer]*, pid*[integer]*)
- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetAssembly(index*[integer]*)
- GetComments()
- GetConnection(index*[integer]*)
- GetParameter(prop*[string]*)
- GetPoint(index*[integer]*)
- GetPointData(rpt*[integer]*)
- GetPointTitle(rpt*[integer]*)
- Next()
- Previous()
- RemoveConnection(index*[integer]*)
- RemoveNodeSetFromAssembly(index*[integer]*, nsid*[integer]*)
- RemovePartFromAssembly(index*[integer]*, pid*[integer]*)
- RemovePartSetFromAssembly(index*[integer]*, psid*[integer]*)
- RemovePoint(index*[integer]*)

- SetConnection(index*[integer]*, data*[object]*)
- SetFlag(flag*[Flag]*)
- SetPoint(index*[integer]*, data*[object]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()

Mechanism constants

Constants for Connection types

Name	Description
Mechanism.COUPLER	Coupler mechanism connection
Mechanism.HINGE	Hinge mechanism connection
Mechanism.LINE	Line mechanism connection
Mechanism.PIN	Pin mechanism connection

Constants for Coupler modes

Name	Description
Mechanism.ROTATION	Rotational coupling on mechanism coupler
Mechanism.TRANSLATION	Translational coupling on mechanism coupler

Mechanism properties

Name	Type	Description
assemblies	integer	Number of assemblies defined. (read only)
connections	integer	Number of connections defined. (read only)
exists	logical	true if mechanism exists, false if referred to but not defined. (read only)
id	integer	Mechanism number. Also see the label property which is an alternative name for this. (read only)
include	integer	The Include file number that the mechanism is in.
label	integer	Mechanism number. Also see the id property which is an alternative name for this. (read only)
model	integer	The Model number that the mechanism is in.
points	integer	Number of reference points defined. (read only)
title	string	Mechanism title.

Detailed Description

The Mechanism class allows you to create, modify, edit and manipulate mechanism cards. See the documentation below for more details.

Details of functions

AddNodeSetToAssembly(index*[integer]*, nsid*[integer]*)

Description

Add node set to assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly in which you want to add node set. **Note that reference points start at 0, not 1**. 0 <= index < assemblies
nsid	integer	The node set ID that you want to add.

Return type

No return value

Example

To add node set 3 in 3rd assembly in mechanism m:

```
m.AddNodeSetToAssembly(2,3);
```

AddPartSetToAssembly(index*[integer]*, psid*[integer]*)

Description

Add part set to assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly in which you want to add part set. **Note that reference points start at 0, not 1**. 0 <= index < assemblies
psid	integer	The part set ID that you want to add.

Return type

No return value

Example

To add part set 3 in 3rd assembly in mechanism m:

```
m.AddPartSetToAssembly(2,3);
```

AddPartToAssembly(index*[integer]*, pid*[integer]*)

Description

Add part to assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly in which you want to add part. **Note that reference points start at 0, not 1**. 0 <= index < assemblies

pid	integer	The part ID that you want to add.

Return type

No return value

Example

To add part 3 in 3rd assembly in mechanism m:

```
m.AddPartToAssembly(2,3);
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a mechanism.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the mechanism

Return type

No return value

Example

To associate comment c to the mechanism m:

```
m.AssociateComment(c);
```

Blank()

Description

Blanks the mechanism

Arguments

No arguments

Return type

No return value

Example

To blank mechanism m:

```
m.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be blanked in

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the mechanisms in model m:

```
Mechanism.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged mechanisms will be blanked in
flag	Flag	Flag set on the mechanisms that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the mechanisms in model m flagged with f:

```
Mechanism.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the mechanism is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if mechanism m is blanked:

```
if (m.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the mechanism.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the mechanism

Return type

No return value

Example

To clear flag f for mechanism m:

```
m.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the mechanism.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

Mechanism object

Example

To copy mechanism m into mechanism z:

```
var z = m.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a mechanism.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the mechanism

Return type

No return value

Example

To detach comment c from the mechanism m:

```
m.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for mechanism. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for mechanism m:

```
m.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first mechanism in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first mechanism in

Return type

Mechanism object (or null if there are no mechanisms in the model).

Example

To get the first mechanism in model m:

```
var m = Mechanism.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free mechanism label in the model. Also see Mechanism.LastFreeLabel(), Mechanism.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free mechanism label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

Mechanism label.

Example

To get the first free mechanism label in model m:

```
var label = Mechanism.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the mechanisms in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be flagged in
flag	Flag	Flag to set on the mechanisms

Return type

No return value

Example

To flag all of the mechanisms with flag f in model m:

```
Mechanism.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the mechanism is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the mechanism

Return type

true if flagged, false if not.

Example

To check if mechanism m has flag f set on it:

```
if (m.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each mechanism in the model.
Note that ForEach has been designed to make looping over mechanisms as fast as possible and so has some limitations.
Firstly, a single temporary Mechanism object is created and on each function call it is updated with the current mechanism data. This means that you should not try to store the Mechanism object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new mechanisms inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms are in
func	function	Function to call for each mechanism
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the mechanisms in model m:

```
Mechanism.ForEach(m, test);
function test(m)
{
// m is Mechanism object
}
```

To call function test for all of the mechanisms in model m with optional object:

```
var data = { x:0, y:0 };
Mechanism.ForEach(m, test, data);
function test(m, extra)
{
// m is Mechanism object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of Mechanism objects for all of the mechanisms in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get mechanisms from

Return type

Array of Mechanism objects

Example

To make an array of Mechanism objects for all of the mechanisms in model m

```
var m = Mechanism.GetAll(m);
```

GetAssembly(index*[integer]*)

Description

Returns the information for an assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly you want the coordinates for. **Note that reference points start at 0, not 1**. 0 <= index < assemblies

Return type

Object with the following properties:

Name	Type	Description
label	integer	Assembly label
parent	integer	Parent assembly label
title	string	Assembly title

Example

To get the information for the 3rd assembly for mechanism m:

```
var info = m.GetAssembly(2);
```

GetComments()

Description

Extracts the comments associated to a mechanism.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the mechanism m:

```
var comm_array = m.GetComments();
```

GetConnection(index*[integer]*)

Description

Returns the information for a connection

Arguments

Name	Type	Description
index	integer	The index of the connection you want the information for. **Note that connections start at 0, not 1**. 0 <= index < connections

Return type

Object with the following properties:

Name	Type	Description
angle	real	Current angle in degrees (for Mechanism.LINE and Mechanism.HINGE)
assembly1	integer	Assembly 1 label
assembly2	integer	Assembly 2 label
assembly3	integer	Assembly 3 label
coefficient1	real	Coefficient for linear coupler equation for connection 1 (for Mechanism.COUPLER)
coefficient2	real	Coefficient for linear coupler equation for connection 2 (for Mechanism.COUPLER)
coefficient3	real	Coefficient for linear coupler equation for connection 3 (for Mechanism.COUPLER)
connection1	integer	Connection 1 label (for Mechanism.COUPLER)
connection2	integer	Connection 2 label (for Mechanism.COUPLER)
connection3	integer	Connection 3 label (for Mechanism.COUPLER)
distance	real	Current distance (for Mechanism.LINE)
factor1	real	Factor 1 on Assembly 3 (Mechanism.LINE only)
factor2	real	Factor 2 on Assembly 3 (Mechanism.LINE only)
label	integer	Connection label
locked	integer	1 if locked (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
mode1	integer	Coupling mode for connection 1. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
mode2	integer	Coupling mode for connection 2. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
mode3	integer	Coupling mode for connection 3. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
node1	integer	Node 1 label (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
node2	integer	Node 2 label (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
nrotation	real	-ve rotation limit in degrees (for Mechanism.LINE and Mechanism.HINGE)
nslide	real	-ve slide translation (for Mechanism.LINE)
protation	real	+ve rotation limit in degrees (for Mechanism.LINE and Mechanism.HINGE)
pslide	real	+ve slide translation (for Mechanism.LINE)
title	string	Connection label
type	integer	Mechanism type (Mechanism.COUPLER, Mechanism.HINGE, Mechanism.LINE, Mechanism.PIN)
x1	real	X1 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
x2	real	X2 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
y1	real	Y1 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
y2	real	Y2 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
z1	real	Z1 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
z2	real	Z2 coordinates (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)

Example

To get the information for the 3rd connection for mechanism m:

```
var info = m.GetConnection(2);
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of Mechanism objects for all of the flagged mechanisms in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get mechanisms from
flag	Flag	Flag set on the mechanisms that you want to retrieve

Return type

Array of Mechanism objects

Example

To make an array of Mechanism objects for all of the mechanisms in model m flagged with f

```
var m = Mechanism.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the Mechanism object for a mechanism ID.

Arguments

Name	Type	Description
Model	Model	Model to find the mechanism in
number	integer	number of the mechanism you want the Mechanism object for

Return type

Mechanism object (or null if mechanism does not exist).

Example

To get the Mechanism object for mechanism 100 in model m

```
var m = Mechanism.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a Mechanism property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the Mechanism.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	mechanism property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if Mechanism property m.example is a parameter:

```
Options.property_parameter_names = true;
if (m.GetParameter(m.example) ) do_something...
Options.property_parameter_names = false;
```

To check if Mechanism property m.example is a parameter by using the GetParameter method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

GetPoint(index*[integer]*)

Description

Returns the information for a reference point

Arguments

Name	Type	Description
index	integer	The index of the reference point you want the information for. **Note that reference points start at 0, not 1**. 0 <= index < points

Return type

Object with the following properties:

Name	Type	Description
assembly	integer	Assembly label
csys	integer	Coordinate system
hpt	boolean	If point has been automatically created by PRIMER at the H-point
label	integer	Point label
node	integer	Node label (0 if coordinate)
rx	boolean	Point restrained rotationally in X
ry	boolean	Point restrained rotationally in Y
rz	boolean	Point restrained rotationally in Z
title	string	Point title
tx	boolean	Point restrained translationally in X
ty	boolean	Point restrained translationally in Y
tz	boolean	Point restrained translationally in Z
x	real	Node/point x coordinate
y	real	Node/point y coordinate
z	real	Node/point z coordinate

Example

To get the information for the 3rd reference point for mechanism m:

```
var info = m.GetPoint(2);
```

GetPointData(rpt*[integer]*)

Description

Returns the coordinates of a reference point

Arguments

Name	Type	Description
rpt	integer	The reference point you want the coordinates for. **Note that reference points start at 0, not 1**.

Return type

Array containing the reference point coordinates

Example

To get the coordinates of the 3rd reference point for mechanism mec:

```
var c = mec.GetPointData(2)
```

GetPointTitle(rpt*[integer]*)

Description

Returns the title of a reference point

Arguments

Name	Type	Description
rpt	integer	The reference point you want the title for. **Note that reference points start at 0, not 1**.

Return type

The reference point title

Example

To get the title of the 3rd reference point for mechanism mec:

```
var c = mec.GetPointTitle(2)
```

Last(Model*[Model]*) [static]

Description

Returns the last mechanism in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last mechanism in

Return type

Mechanism object (or null if there are no mechanisms in the model).

Example

To get the last mechanism in model m:

```
var m = Mechanism.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free mechanism label in the model. Also see Mechanism.FirstFreeLabel(), Mechanism.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free mechanism label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

Mechanism label.

Example

To get the last free mechanism label in model m:

```
var label = Mechanism.LastFreeLabel(m);
```

Next()

Description

Returns the next mechanism in the model.

Arguments

No arguments

Return type

Mechanism object (or null if there are no more mechanisms in the model).

Example

To get the mechanism in model m after mechanism m:

```
var m = m.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) mechanism label in the model. Also see Mechanism.FirstFreeLabel(), Mechanism.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free mechanism label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

Mechanism label.

Example

To get the next free mechanism label in model m:

```
var label = Mechanism.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a mechanism.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only mechanisms from that model can be picked. If the argument is a Flag then only mechanisms that are flagged with *limit* can be selected. If omitted, or null, any mechanisms from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

Mechanism object (or null if not picked)

Example

To pick a mechanism from model m giving the prompt 'Pick mechanism from screen':

```
var m = Mechanism.Pick('Pick mechanism from screen', m);
```

Previous()

Description

Returns the previous mechanism in the model.

Arguments

No arguments

Return type

Mechanism object (or null if there are no more mechanisms in the model).

Example

To get the mechanism in model m before mechanism m:

```
var m = m.Previous();
```

RemoveConnection(index*[integer]*)

Description

Removes a connection from a mechanism

Arguments

Name	Type	Description
index	integer	The index of the connection you want to remove. **Note that connections start at 0, not 1**. 0 <= index < connections

Return type

no return value

Example

To remove the 3rd connection for mechanism m:

```
m.RemoveConnection(2);
```

RemoveNodeSetFromAssembly(index*[integer]*, nsid*[integer]*)

Description

Remove node set from assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly from which you want to remove the node set. **Note that reference points start at 0, not 1**. 0 <= index < assemblies
nsid	integer	The node set ID that you want to remove.

Return type

No return value

Example

To remove node set 3 from 3rd assembly in mechanism m:

```
m.RemoveNodeSetFromAssembly(2,3);
```

RemovePartFromAssembly(index*[integer]*, pid*[integer]*)

Description

Remove part from assembly

Arguments

Name	Type	Description

index	integer	The index of the assembly from which you want to remove the part. **Note that reference points start at 0, not 1**. 0 <= index < assemblies
pid	integer	The part ID that you want to remove.

Return type

No return value

Example

To remove part 3 from 3rd assembly in mechanism m:

```
m.RemovePartFromAssembly(2,3);
```

RemovePartSetFromAssembly(index*[integer]*, psid*[integer]*)

Description

Remove part set from assembly

Arguments

Name	Type	Description
index	integer	The index of the assembly from which you want to remove the part set. **Note that reference points start at 0, not 1**. 0 <= index < assemblies
psid	integer	The part set ID that you want to remove.

Return type

No return value

Example

To remove part set 3 from 3rd assembly in mechanism m:

```
m.RemovePartSetFromAssembly(2,3);
```

RemovePoint(index*[integer]*)

Description

Removes a reference point from a mechanism

Arguments

Name	Type	Description
index	integer	The index of the reference point you want to remove. **Note that reference points start at 0, not 1**. 0 <= index < points

Return type

no return value

Example

To remove the 3rd reference point for mechanism m:

```
m.RemovePoint(2);
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the mechanisms in model m, from 1000000:

```
Mechanism.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged mechanisms will be renumbered in
flag	Flag	Flag set on the mechanisms that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the mechanisms in model m flagged with f, from 1000000:

```
Mechanism.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select mechanisms using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting mechanisms
prompt	string	Text to display as a prompt to the user

limit (optional)	Model or Flag	If the argument is a Model then only mechanisms from that model can be selected. If the argument is a Flag then only mechanisms that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any mechanisms can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of mechanisms selected or null if menu cancelled

Example

To select mechanisms from model m, flagging those selected with flag f, giving the prompt 'Select mechanisms':

```
Mechanism.Select(f, 'Select mechanisms', m);
```

To select mechanisms, flagging those selected with flag f but limiting selection to mechanisms flagged with flag l, giving the prompt 'Select mechanisms':

```
Mechanism.Select(f, 'Select mechanisms', l);
```

SetConnection(index*[integer]*, data*[object]*)

Description

Sets the data for a connection in a mechanism

Arguments

Name	Type	Description
index	integer	The index of the connection you want to set. **Note that connections start at 0, not 1**. To add a new connection use index connections

data | object

Name	Type	Description
angle (optional)	real	Current angle in degrees (for Mechanism.LINE and Mechanism.HINGE)
assembly1 (optional)	integer	Assembly 1 label (required for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
assembly2 (optional)	integer	Assembly 2 label (required for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
assembly3	integer	Assembly 3 label (required for Mechanism.LINE)
coefficient1 (optional)	real	Coefficient for linear coupler equation for connection 1 (for Mechanism.COUPLER)
coefficient2 (optional)	real	Coefficient for linear coupler equation for connection 2 (for Mechanism.COUPLER)
coefficient3 (optional)	real	Coefficient for linear coupler equation for connection 3 (for Mechanism.COUPLER)
connection1 (optional)	integer	Connection 1 label (for Mechanism.COUPLER)
connection2 (optional)	integer	Connection 2 label (for Mechanism.COUPLER)
connection3 (optional)	integer	Connection 3 label (for Mechanism.COUPLER)
distance (optional)	real	Current distance (for Mechanism.LINE)
factor1 (optional)	real	Factor 1 on Assembly 3 (Mechanism.LINE only)
factor2 (optional)	real	Factor 2 on Assembly 3 (Mechanism.LINE only)
locked (optional)	integer	1 if locked (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE)
mode1 (optional)	integer	Coupling mode for connection 1. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
mode2 (optional)	integer	Coupling mode for connection 2. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
mode3 (optional)	integer	Coupling mode for connection 3. 0 = translational coupling, 1 = rotational coupling (for Mechanism.COUPLER)
node1 (optional)	integer	Node 1 label (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using x1, y1 and z1)
node2 (optional)	integer	Node 2 label (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using x2, y2 and z2)
nrotation (optional)	real	-ve rotation limit in degrees (for Mechanism.LINE and Mechanism.HINGE)
nslide (optional)	real	-ve slide translation (for Mechanism.LINE)
protation (optional)	real	+ve rotation limit in degrees (for Mechanism.LINE and Mechanism.HINGE)
pslide (optional)	real	+ve slide translation (for Mechanism.LINE)
title (optional)	string	Title
type	integer	Connection type. Can be one of: Mechanism.PIN, Mechanism.LINE, Mechanism.HINGE or Mechanism.COUPLER]
x1 (optional)	real	x1 coordinate (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using node1)
x2 (optional)	real	x2 coordinate (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using node2)
y1 (optional)	real	y1 coordinate (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using node1)
y2 (optional)	real	y2 coordinate (for Mechanism.LINE, Mechanism.PIN and Mechanism.HINGE, not required if using node2)
z1 (optional)	real	z1 coordinate (for Mechanism.LINE, Mechanism.PIN and

Return type

no return value

Example

To add a new pin connection to mechanism m between assemblies 5 and 6 at node 1000 with title "Example connection":

```
var data = { type:Mechanism.PIN, assembly1:5, assembly2:6, node1:1000,
title:"Example connection" };
m.SetConnection(m.connections, data);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the mechanism.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the mechanism

Return type

No return value

Example

To set flag f for mechanism m:

```
m.SetFlag(f);
```

SetPoint(index*[integer]*, data*[object]*)

Description

Sets the data for a reference point in a mechanism

Arguments

Name	Type	Description
index	integer	The index of the reference point you want to set. **Note that reference points start at 0, not 1**. To add a new point use index points

<table>
<tr>
<td>data</td>
<td>object</td>
<td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>assembly</td><td>integer</td><td>Assembly label</td></tr>
<tr><td>csys (optional)</td><td>integer</td><td>Coordinate system label</td></tr>
<tr><td>node (optional)</td><td>integer</td><td>Node label (not required if using x, y and z)</td></tr>
<tr><td>rx (optional)</td><td>boolean</td><td>Point restrained rotationally in X</td></tr>
<tr><td>ry (optional)</td><td>boolean</td><td>Point restrained rotationally in Y</td></tr>
<tr><td>rz (optional)</td><td>boolean</td><td>Point restrained rotationally in Z</td></tr>
<tr><td>title (optional)</td><td>string</td><td>Point title</td></tr>
<tr><td>tx (optional)</td><td>boolean</td><td>Point restrained translationally in X</td></tr>
<tr><td>ty (optional)</td><td>boolean</td><td>Point restrained translationally in Y</td></tr>
<tr><td>tz (optional)</td><td>boolean</td><td>Point restrained translationally in Z</td></tr>
<tr><td>x (optional)</td><td>real</td><td>x coordinate (not required if using node)</td></tr>
<tr><td>y (optional)</td><td>real</td><td>y coordinate (not required if using node)</td></tr>
<tr><td>z (optional)</td><td>real</td><td>z coordinate (not required if using node)</td></tr>
</table>
Object containing the reference point data. The properties can be: Object has the following properties:
</td>
</tr>
</table>

Return type

no return value

Example

To add a new reference point to mechanism m assembly 5 at node 1000 with title "Example point" restrained in x:

```
var data = { assembly:5, node:1000, title:"Example point", tx:true };
m.SetPoint(m.points, data);
```

To add a new reference point to mechanism m assembly 5 at (10, 20, 30) with title "Example point":

```
var data = { assembly:5, x:10, y:20, z:30, title:"Example point" };
m.SetPoint(m.points, data);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the mechanism. The mechanism will be sketched until you either call Mechanism.Unsketch(), Mechanism.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the mechanism is sketched. If omitted redraw is true. If you want to sketch several mechanisms and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch mechanism m:

```
m.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged mechanisms in the model. The mechanisms will be sketched until you either call Mechanism.Unsketch(), Mechanism.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged mechanisms will be sketched in
flag	Flag	Flag set on the mechanisms that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the mechanisms are sketched. If omitted redraw is true. If you want to sketch flagged mechanisms several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all mechanisms flagged with flag in model m:

```
Mechanism.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing mechanisms should be counted. If false or omitted referenced but undefined mechanisms will also be included in the total.

Return type

number of mechanisms

Example

To get the total number of mechanisms in model m:

```
var total = Mechanism.Total(m);
```

Unblank()

Description

Unblanks the mechanism

Arguments

No arguments

Return type

No return value

Example

To unblank mechanism m:

```
m.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the mechanisms in model m:

```
Mechanism.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged mechanisms will be unblanked in
flag	Flag	Flag set on the mechanisms that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the mechanisms in model m flagged with f:

```
Mechanism.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all mechanisms will be unset in
flag	Flag	Flag to unset on the mechanisms

Return type

No return value

Example

To unset the flag f on all the mechanisms in model m:

```
Mechanism.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the mechanism.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the mechanism is unsketched. If omitted redraw is true. If you want to unsketch several mechanisms and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch mechanism m:

```
m.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all mechanisms.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the mechanisms are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all mechanisms in model m:

```
Mechanism.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged mechanisms in the model.

Arguments

Name	Type	Description
Model	Model	Model that all mechanisms will be unsketched in
flag	Flag	Flag set on the mechanisms that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the mechanisms are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all mechanisms flagged with flag in model m:

```
Mechanism.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

Mechanism object.

Example

To check if Mechanism property m.example is a parameter by using the Mechanism.GetParameter() method:

```
if (m.ViewParameters().GetParameter(m.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for mechanism. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for mechanism m:

```
m.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this mechanism.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for mechanism m:

```
var xrefs = m.Xrefs();
```

Model class

The Model class gives you access to models in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll()
- DeleteAll()
- First()
- FirstFreeItemLabel(type*[string]*, layer (optional)*[Include number]*)
- GetAll()
- GetFromID(model number*[integer]*)
- Last()
- LastFreeItemLabel(type*[string]*, layer (optional)*[Include number]*)
- Merge(Master Model*[Model]*, Slave Model*[Model]*, Option to fix clashes (optional)*[constant]*, Merge nodes flag (optional)*[boolean]*, dist (required if merge nodes flag used) (optional)*[real]*, label (optional)*[integer]*, position (optional)*[integer]*)
- NextFreeItemLabel(type*[string]*, layer (optional)*[Include number]*)
- Read(filename*[string]*, filetype (optional)*[constant]*, number (optional)*[integer]*)
- Select(prompt*[string]*, modal (optional)*[boolean]*)
- Total()
- UnblankAll()

Member functions

- AreaVolumeFlagged(flag*[Flag]*)
- Attached(flag*[Flag]*, redraw (optional)*[boolean]*)
- Autofix()
- Blank()
- BlankFlagged(flag*[Flag]*, redraw (optional)*[boolean]*)
- CentreOfGravity()
- Check(filename*[string]*, detailed (optional)*[boolean]*, json (optional)*[boolean]*, include (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(update (optional)*[boolean]*)
- CopyFlagged(flag*[Flag]*, update (optional)*[boolean]*)
- Delete()
- DeleteFlagged(flag*[Flag]*, recursive (optional)*[boolean]*)
- DeleteInclude(Include label*[integer]*, method (optional)*[constant]*, force (optional)*[boolean]*)
- FindElemEnd()
- FindElemInit(flag (optional)*[Flag]*)
- FlagDuplicate(flag*[Flag]*)
- GetIncludeTransformOffsets()
- Hide()
- Import(filename*[string]*)
- ImportInclude(source*[String OR Include Object]*, target (optional)*[Include Object]*)
- ImportIncludeTransform(filename*[string]*, idnoff*[integer]*, ideoff*[integer]*, idpoff*[integer]*, idmoff*[integer]*, idsoff*[integer]*, idfoff*[integer]*, iddoff*[integer]*, idroff*[integer]*)
- Mass()
- MassPropCalc(flag*[Flag]*)
- MergeNodes(flag*[Flag]*, dist*[real]*, label (optional)*[integer]*, position (optional)*[integer]*)
- PopulateInitialVelocities()
- PropagateFlag(flag*[Flag]*)
- RenumberAll(start*[integer]*)
- RenumberFlagged(flag*[Flag]*, start*[integer]*, mode (optional)*[constant]*)
- SetColour(colour*[colour from Colour class.]*)
- SetFlag(flag*[Flag]*)
- Show()

- Unblank()
- UnblankFlagged(flag*[Flag]*, redraw (optional)*[boolean]*)
- UnsketchAll(redraw (optional)*[boolean]*)
- UpdateGraphics()
- UsesLargeLabels()
- Write(filename*[string]*, options (optional)*[object]*)
- Write(filename*[string]*, method (optional)*[constant]*, path (optional)*[constant]*, separator (optional)*[constant]*, version (optional)*[string]*, large (optional)*[boolean]*) [deprecated]

Model constants

Constants for compress mode

Name	Description
Model.INDIVIDUAL_GZIP	Each file 'name.key' is 'gzipped' to become the individual file 'name.key.gz'
Model.INDIVIDUAL_ZIP	Each file 'name.key' is 'zipped' to become the individual file 'name.key.zip'
Model.KEEP_ORIGINAL	Each file 'name.key' is written using its original compression: uncompressed, '.gz. or '.zip' format
Model.PACKAGED_ZIP	Suitable for models with include files where the entire model is packed into a single .zip file, preserving its directory structure.

Constants for compress switch

Name	Description
Model.COMPRESS_KEEP	Keeps the keyout compression format same as that of what was read in.
Model.COMPRESS_OFF	Switches off compression during keyout.
Model.COMPRESS_ON	Switches on compression during keyout.

Constants for filetype

Name	Description
Model.ABAQUS	ABAQUS input file
Model.IGES	IGES 5.3 geometry file
Model.LSDYNA	LS-DYNA keyword file
Model.NASTRAN	NASTRAN bulk data file
Model.RADIOSS	RADIOSS block format file

Constants for include deletion

Name	Description
Model.REMOVE_FROM_SETS	Only deletes items within the include selected but may remove items from sets in other includes.
Model.REMOVE_INCLUDE_ONLY	Only deletes items within the include selected without removing items from sets in other includes.
Model.REMOVE_JUNIOR	Delete items in other includes if they 'belong' to items in this include file but are considered to be 'junior' in the standard PRIMER hierarchy.

Constants for mass_properties_calculation

Name	Description

Model.CENTRE_AT_COFG	Uses the centre at centre of gravity in calculation of inertia properties.
Model.GLOBAL_AXES	GLOBAL AXES
Model.LOCAL_AXES	LOCAL AXES
Model.PRINCIPAL_AXES	PRINCIPAL AXES
Model.USER_DEFINED_CENTRE	Uses the user defined centre in calculation of inertia properties.

Constants for merge

Name	Description
Model.DISCARD_MASTER_CLASH	Merge option - discard master items only on clash
Model.DISCARD_SLAVE_CLASH	Merge option - discard slave items only on clash
Model.INCREASE_MASTER_ALWAYS	Merge option - increase master items always
Model.INCREASE_MASTER_CLASH	Merge option - increase master items only on clash
Model.INCREASE_SLAVE_ALWAYS	Merge option - increase slave items always
Model.INCREASE_SLAVE_CLASH	Merge option - increase slave items only on clash

Constants for renumber

Name	Description
Model.IGNORE_CLASH	Renumber option - Ignore clashes.
Model.MOVE_CLASH_UP	Renumber option - Move clashing > higest label.
Model.RENUMBER_TO_FREE	Renumber option - Renumber to next free label.
Model.SHIFT_ALL_UP	Renumber option - Shift upwards to make space.

Constants for write hook

Name	Description
Model.WRITE_DIALOGUE	Flag that triggers the write hook script from the dialogue box
Model.WRITE_INCLUDE_TREE	Flag that triggers the write hook script from the include file tree
Model.WRITE_MODEL	Flag that triggers the write hook script from the model tab
Model.WRITE_SELECT_INCLUDES	Flag that triggers the write hook script from the select include files panel

Model class properties

Name	Type	Description
highest	integer	The highest model number present in PRIMER

Model properties

Name	Type	Description
binary (read only)	boolean	If model is in binary then it will be 1(true) else 0(false).
comments	string	Comments stored at the top of the master model file.
compress (read only)	boolean	If model is compressed then it will be 1(true) else 0(false).

compressMode (read only)	integer	This option can be used to know the mode of compression. Can be Model.INDIVIDUAL_GZIP or Model.INDIVIDUAL_ZIP or Model.PACKAGED_ZIP
control (read only)	Control object	Control cards for model. See Control for more details.
damping (read only)	Damping object	Damping cards for model. See Damping for more details.
database (read only)	Database object	Database cards for model. See Database for more details.
fileStartAscii (read only)	boolean	If the beginning of the master file (*CONTROL etc) is in ascii then 1(true) else 0(false)(NOTE: If master file is ascii then fileStartAscii won't be checked and show 0(false)).
filename (read only)	string	Name of file that model was read from (blank if model created)
id	logical	If ID flag set for *KEYWORD card
layer	integer	The current layer for the model. This is the label of the Include file or 0 for the main file. See also Include.MakeCurrentLayer()
loadBody (read only)	LoadBody object	LoadBody cards for model. See LoadBody for more details.
masterAscii (read only)	boolean	If master file is in ascii then 1(true) else 0(false).
num	string	Model num (for _ID)
number	integer	Model number
path (read only)	string	Path that model was read from (blank if model created)
project	string	Model project (for _ID)
readlog	string	Full path of the readlog file
stage	string	Model stage (for _ID)
title	string	Model title
visible	logical	Model visibility flag

Detailed Description

The Model class allows you to do various operations on models in PRIMER. There are various methods available that allow you do create, read, blank models etc. See the documentation below for more details.

Constructor

new Model(number (optional)*[integer]*)

Description

Create a new model in PRIMER

Arguments

Name	Type	Description
number (optional)	integer	Model number to create. If omitted the next free model number will be used.

Return type

Model object

Example

To create a new model

```
var m = new Model();
```

To create model 10

```
var m = new Model(10);
```

Details of functions

AreaVolumeFlagged(flag*[Flag]*)

Description

Calculates the Area/Volume of the selected items.
Note: The area calculation is based only on shell elements, and the volume calculation is based only on solid elements.

Arguments

Name	Type	Description
flag	Flag	Flag set on entities you wish to calculate area/volume for

Return type

Object with the following properties:

Name	Type	Description
area	real	Area of flagged items
volume	real	Volume of flagged items

Example

To calculate the area/volume properties of the items flagged by flag f.

```
var props = m.AreaVolumeFlagged(f);
var area = props.area;
var volume = props.volume;
```

Attached(flag*[Flag]*, redraw (optional)*[boolean]*)

Description

Finds attached items to flagged items. The attached items are flagged with the same flag.

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to find attached to
redraw (optional)	boolean	If true, the display will be updated to display only the original flagged items and the attached items.

Return type

No return value

Example

To find items attached to items flagged with flag f in model m:

```
m.Attached(f);
```

Autofix()

Description

Autofix option does a model check and autofixes all the fixable errors in the model

Arguments

No arguments

Return type

No return value

Example

To autofix fixable errors of the model ’m’

```
m.Autofix();
```

Blank()

Description

Blanks a model in PRIMER

Arguments

No arguments

Return type

No return value

Example

To blank model object m

```
m.Blank();
```

BlankAll() [static]

Description

Blanks all models

Arguments

No arguments

Return type

No return value

Example

To blank all models

```
Model.BlankAll();
```

BlankFlagged(flag*[Flag]*, redraw (optional)*[boolean]*)

Description

Blanks all of the flagged items in the model.

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank everything in model m flagged with flag f:

```
m.BlankFlagged(f);
```

CentreOfGravity()

Description

Returns the centre of gravity for a model

Arguments

No arguments

Return type

An array containing the x, y and z coordinates for the CofG.

Example

To get the centre of gravity for model m:

```
var cofg = m.CentreOfGravity();
var x = cofg[0];
var y = cofg[1];
var z = cofg[2];
```

Check(filename*[string]*, detailed (optional)*[boolean]*, json (optional)*[boolean]*, include (optional)*[boolean]*)

Description

Checks a model, writing any errors to file.

Arguments

Name	Type	Description
filename	string	Name of file to write errors to
detailed (optional)	boolean	If set to "true", detailed error messages are given.

json (optional)	boolean	If set, output in filename will be written in JSON format. If omitted json will be set to false. If JSON format is written then detailed will automatically be set. Note that when writing JSON format the labels produced can be strings instead of integers in some rare cases. If you are writing a script to read a JSON file, it must be able to cope with this. Specifically if the item is a character label the label will be a string. For child collect sets the label will be a string of the format 'X_Y' where X is the parent set label and Y will be the child set number (1, 2, 3 ...). In this case use Set.GetCollectChild() to get the object.
include (optional)	boolean	If set, error messages will be written in include by include layout. This option is not applicable if JSON is set.

Return type

No return value

Example

To check model m, writing detailed errors to file 'errors.txt' in include layout:

```
m.Check('errors.txt', true, false, true);
```

To check a model writing the warnings/errors as JSON to file 'errors.json', parse it and write them to the dialogue box:

```
m.Check('errors.json', true, true);
var f = new File('errors.json', File.READ);
var json = f.ReadAll();
f.Close();
var o = JSON.parse(json);
for (var e in o)                                    // "error" or "warning"
{
    Message(e);
    for (var t in o[e])                             // type
    {
        Message("  "+t);
        for (var m in o[e][t])                      // message
        {
            Message("    "+m);
            for (var i=0; i<o[e][t][m].length; i++)  // Array of objects
containing label and include
            {
                Message("        "+o[e][t][m][i].label+" (include
"+o[e][t][m][i].include+")");
            }
        }
    }
}
```

ClearFlag(flag[*Flag*])

Description

Clears the flagging for a model in PRIMER. See also Model.PropagateFlag(), Model.SetFlag(), global.AllocateFlag() and global.ReturnFlag().

Arguments

Name	Type	Description
flag	Flag	Flag to clear

Return type

No return value

Example

To clear flag f for everything in model m:

```
m.ClearFlag(f);
```

Copy(update (optional)*[boolean]*)

Description

Copy model to the next free model in PRIMER

Arguments

Name	Type	Description
update (optional)	boolean	If the graphics should be updated after the model is copied. If omitted update will be set to false

Return type

Model object for new model.

Example

To copy model m to the next free model in PRIMER.

```
var mnew = m.Copy();
```

CopyFlagged(flag*[Flag]*, update (optional)*[boolean]*)

Description

Copy flagged items in a model to the next free model in PRIMER

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to copy
update (optional)	boolean	If the graphics should be updated after the model is copied. If omitted update will be set to false

Return type

Model object for new model.

Example

To copy everything in model m flagged with flag f to the next free model in PRIMER.

```
var mnew = m.CopyFlagged(f);
```

Delete()

Description

Deletes a model in PRIMER
Do not use the Model object after calling this method.

Arguments

No arguments

Return type

No return value

Example

To delete model m in PRIMER

```
m.Delete();
```

DeleteAll() [static]

Description

Deletes all existing models from PRIMER

Arguments

No arguments

Return type

No return value

Example

To delete all models

```
Model.DeleteAll();
```

DeleteFlagged(flag*[Flag]*, recursive (optional)*[boolean]*)

Description

Deletes all of the flagged items in the model. Note that this may not actually delete all of the items. For example if a node is flagged but the node is used in a shell which is not flagged then the node will not be deleted.

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to delete
recursive (optional)	boolean	If deletion is recursive (for example, if a shell is deleted with recursion on the shell nodes will be deleted if possible). If omitted recursive will be set to true.

Return type

No return value

Example

To delete everything in model m flagged with flag f:

```
m.DeleteFlagged(f);
```

DeleteInclude(Include label*[integer]*, method (optional)*[constant]*, force (optional)*[boolean]*)

Description

Tries to delete an include file from the model. Note that this may not actually delete the include file. For example if some of the items in the include file are required by other things in different includes then the include file will not be deleted.

Arguments

Name	Type	Description

Include label	integer	label of include file that you want to delete
method (optional)	constant	Method for deleting items. Must be Model.REMOVE_FROM_SETS (default), Model.REMOVE_JUNIOR or Model.REMOVE_INCLUDE_ONLY. Model.REMOVE_FROM_SETS will only delete items within the include selected but may remove items from sets in other includes. Model.REMOVE_JUNIOR may delete items in other includes - this will happen if they 'belong' to items in this include and are considered 'junior' Model.REMOVE_INCLUDE_ONLY does the same as Model.REMOVE_FROM_SETS but will **not** remove items from sets in other includes.
force (optional)	boolean	Forcible deletion option (for example, a node is deleted even when it is referenced by a shell which is not deleted). This will remove the include file (not just the contents) from the model. If this argument is omitted, force will be set to false.

Return type

true if include successfully deleted, false otherwise

Example

To delete include file number 5 in model m removing items from sets in other includes if required:

```
m.DeleteInclude(5, 1);
```

FindElemEnd()

Description

Tidy memory allocation incurred by function which finds elements within a box. See also Model.FindElemInit()

Arguments

No arguments

Return type

No return value

Example

```
m.FindElemEnd();
```

FindElemInit(flag (optional)*[Flag]*)

Description

Initialize setup so that all flagged beams, shells, solids, tshells in model can be tested to see if they are within box. See also Shell.FindShellInBox() Tshell.FindTshellInBox() Solid.FindSolidInBox() Beam.FindBeamInBox() To return memory Model.FindElemEnd()

Arguments

Name	Type	Description
flag (optional)	Flag	If omitted all shells, solids, tshells, beams will be considered.

Return type

No return value

Example

To initialize find setup for elements flagged with f in model m:

```
m.FindElemInit(f);
```

First() [static]

Description

Returns the Model object for the first model in PRIMER (or null if there are no models)

Arguments

No arguments

Return type

Model object

Example

To get the Model object for the first model:

```
var m = Model.First();
```

FirstFreeItemLabel(type*[string]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free label for an item type in the model. Also see Model.LastFreeItemLabel() and Model.NextFreeItemLabel().

Arguments

Name	Type	Description
type	string	The type of the item (for a list of types see Appendix I of the PRIMER manual).
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

integer

Example

To get the first free node label in model m:

```
var label = m.FirstFreeItemLabel("NODE");
```

FlagDuplicate(flag*[Flag]*)

Description

Flag all nodes referenced in two different includes

Arguments

Name	Type	Description
flag	Flag	Flag which will be used to flag the "duplicate" nodes

Return type

No return value

Example

To Flag with flag f all the nodes referenced in different includes from model m

```
m.FlagDuplicate(f);
```

GetAll() [static]

Description

Returns an array of Model objects for all the models in PRIMER

Arguments

No arguments

Return type

Array of Model objects

Example

To make an array of Model objects for all of the models in PRIMER

```
var m = Model.GetAll();
```

GetFromID(model number*[integer]*) [static]

Description

Returns the Model object for a model ID or null if model does not exist

Arguments

Name	Type	Description
model number	integer	number of the model you want the Model object for

Return type

Model object

Example

To get the Model object for model number 1

```
var m = Model.GetFromID(1);
```

GetIncludeTransformOffsets()

Description

Looks at all of the items in the model and determines values for IDNOFF, IDEOFF, IDPOFF etc that could be used with Model.ImportIncludeTransform to guarantee that there would not be any clashes with existing items in the model.

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
iddoff	integer	Offset to define id
ideoff	integer	Offset to element id
idfoff	integer	Offset to function id
idmoff	integer	Offset to material id
idnoff	integer	Offset to node id
idpoff	integer	Offset to part id
idroff	integer	Offset to section, hourglass, EOS id
idsoff	integer	Offset to set id

Example

To determine offsets for model m and then import an include transform "test.inc":

```
var o = m.GetIncludeTransformOffsets();
if (o)
{
    var success = m.ImportIncludeTransform("test.inc", o.idnoff, o.ideoff,
o.idpoff, o.idmoff, o.idsoff, o.idfoff, o.iddoff, o.idroff);
}
```

Hide()

Description

Hides a model in PRIMER

Arguments

No arguments

Return type

No return value

Example

To hide model m in PRIMER

```
m.Hide();
```

Import(filename*[string]*)

Description

Imports a file into model m. The model can already contain items. However, **note that if the file cannot be imported because of a label clash or other problem PRIMER may delete the model and the script will terminate**. Note prior to v17 of PRIMER imported data would always be imported to the master model, irrespective of the current layer. From v17 onwards this has been corrected and the current layer is used to determine the destination of imported data.

Arguments

Name	Type	Description
filename	string	Filename of the LS-DYNA keyword file you want to import

Return type

0: No errors/warnings.
> 0: This number of errors occurred.
< 0: Absolute number is the number of warnings that occurred.

Example

To import file "test.key" into model m

```
m.Import("test.key");
```

ImportInclude(source*[String OR Include Object]*, target (optional)*[Include Object]*)

Description

Imports a keyword file or an Include object from different model as a new include or into an existing include file for model m. The labels of any items in the imported include contents that clash with existing labels will automatically be renumbered with one exception. The behaviour for *SET_COLLECT cards can be controlled with Options.merge_set_collect.

Arguments

Name	Type	Description
source	String OR Include Object	Can either be a Filename of the LS-DYNA include file you want to import, OR Include object of another model you want to import
target (optional)	Include Object	Include file object of current model if the Import has to be done in an existing include .

Return type

Include object for include file

Example

To import include file "include.key" into model m:

```
m.ImportInclude("include.key");
```

To import include file "include.key" into existing include incl in model m:

```
m.ImportInclude("include.key", incl);
```

To import include number 5 from model m2 into model m:

```
var incl = Include.GetFromID(m2, 5);
m.ImportInclude(incl);
```

ImportIncludeTransform(filename*[string]*, idnoff*[integer]*, ideoff*[integer]*, idpoff*[integer]*, idmoff*[integer]*, idsoff*[integer]*, idfoff*[integer]*, iddoff*[integer]*, idroff*[integer]*)

Description

Imports a file as an include transform file for model m. The labels of any items in the include file will be renumbered by idnoff, ideoff etc.

Arguments

Name	Type	Description
filename	string	Filename of the LS-DYNA include file you want to import

idnoff	integer	Offset for nodes in the file
ideoff	integer	Offset for elements in the file
idpoff	integer	Offset for parts in the file
idmoff	integer	Offset for materials in the file
idsoff	integer	Offset for sets in the file
idfoff	integer	Offset for functions and tables in the file
iddoff	integer	Offset for defines in the file
idroff	integer	Offset for other labels in the file

Return type

Include object if successful, null if not

Example

To import include transform file "include.key" into model m using 1000 for all offsets

```
m.ImportIncludeTransform("include.key", 1000, 1000, 1000, 1000, 1000, 1000,
1000, 1000);
```

Last() [static]

Description

Returns the Model object for the last model in PRIMER (or null if there are no models)

Arguments

No arguments

Return type

Model object

Example

To get the Model object for the last model:

```
var m = Model.Last();
```

LastFreeItemLabel(type*[string]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free label for an item type in the model. Also see Model.FirstFreeItemLabel() and Model.NextFreeItemLabel().

Arguments

Name	Type	Description
type	string	The type of the item (for a list of types see Appendix I of the PRIMER manual).
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

integer

Example

To get the last free node label in model m:

```
var label = m.LastFreeItemLabel("NODE");
```

Mass()

Description

Returns the mass for a model

Arguments

No arguments

Return type

real

Example

To get the mass for model m:

```
var mass = m.Mass();
```

MassPropCalc(flag*[Flag]*)

Description

Calculates the Mass, CoG, and Intertia Tensor of the flagged items and returns an object with the above properties. See Properties for mass properties calculation under options class to configure inclusion of lumped mass, etc.

Arguments

Name	Type	Description
flag	Flag	Calculate mass propetries of flagged items

Return type

Object with the following properties:

Name	Type	Description
cofgx	real	X coordinate of centre of gravity
cofgy	real	Y coordinate of centre of gravity
cofgz	real	Z coordinate of centre of gravity
inerxx	real	XX inertia
inerxy	real	XY inertia
inerxz	real	XZ inertia
ineryy	real	YY inertia
ineryz	real	YZ inertia
inerzz	real	ZZ inertia
mass	real	Mass

Example

To calculate the mass properties of the items flagged by flag f

```
var props = m.MassPropCalc(f);
```

Merge(Master Model*[Model]*, Slave Model*[Model]*, Option to fix clashes (optional)*[constant]*, Merge nodes flag (optional)*[boolean]*, dist (required if merge nodes flag used) (optional)*[real]*, label (optional)*[integer]*, position (optional)*[integer]*) [static]

Description

Merge 2 models together to make a new model.

Arguments

Name	Type	Description
Master Model	Model	Master Model for merge.
Slave Model	Model	Slave Model for merge.
Option to fix clashes (optional)	constant	Type of fix. Can be Model.INCREASE_SLAVE_ALWAYS, Model.INCREASE_SLAVE_CLASH, Model.DISCARD_SLAVE_CLASH, Model.INCREASE_MASTER_ALWAYS, Model.INCREASE_MASTER_CLASH or Model.DISCARD_MASTER_CLASH
Merge nodes flag (optional)	boolean	If this flag is set to true, Primer will merge nodes after the model merge.
dist (required if merge nodes flag used) (optional)	real	Nodes closer than dist will be potentially merged.
label (optional)	integer	Label to keep after merge. If > 0 then highest label kept. If <= 0 then lowest kept. If omitted the lowest label will be kept.
position (optional)	integer	Position to merge at. If > 0 then merged at highest label position. If < 0 then merged at lowest label position. If 0 then merged at midpoint. If omitted the merge will be done at the lowest label.

Return type

Model object (or null if the merge is unsuccessful)

Example

To merge models m1 and m2 together:

```
var m = Model.Merge(m1, m2);
```

MergeNodes(flag*[Flag]*, dist*[real]*, label (optional)*[integer]*, position (optional)*[integer]*)

Description

Attempts to merge nodes on items flagged with flag for this model in PRIMER. Merging nodes on *AIRBAG_SHELL_REFERENCE_GEOMETRY can be controlled by using Options.node_replace_asrg. Also see Node.Merge().

Arguments

Name	Type	Description

flag	Flag	Flag set on items to merge nodes
dist	real	Nodes closer than dist will be potentially merged.
label (optional)	integer	Label to keep after merge. If > 0 then highest label kept. If <= 0 then lowest kept. If omitted the lowest label will be kept.
position (optional)	integer	Position to merge at. If > 0 then merged at highest label position. If < 0 then merged at lowest label position. If 0 then merged at midpoint. If omitted the merge will be done at the lowest label.

Return type

The number of nodes merged

Example

To (try to) merge nodes on everything in model m flagged with flag f, with a distance of 0.1:

```
m.MergeNodes(f, 0.1);
```

NextFreeItemLabel(type*[string]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free label for an item type in the model. Also see Model.FirstFreeItemLabel() and Model.LastFreeItemLabel().

Arguments

Name	Type	Description
type	string	The type of the item (for a list of types see Appendix I of the PRIMER manual).
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

integer

Example

To get the next free node label in model m:

```
var label = m.NextFreeItemLabel("NODE");
```

PopulateInitialVelocities()

Description

Populate the initial velocity field (nvels) for all nodes of the model

Arguments

No arguments

Return type

No return value

Example

```
m.PopulateInitialVelocities();
```

PropagateFlag(flag*[Flag]*)

Description

Propagates the flagging for a model in PRIMER. For example if a part in the model is flagged, this will flag the elements in the part, the nodes on those elements... See also Model.ClearFlag(), Model.SetFlag(), global.AllocateFlag() and global.ReturnFlag().

Arguments

Name	Type	Description
flag	Flag	Flag to propagate

Return type

No return value

Example

To propagate the flagging in model m for flag f

```
m.PropagateFlag(f);
```

Read(filename*[string]*, filetype (optional)*[constant]*, number (optional)*[integer]*) [static]

Description

Reads a file into the first free model in PRIMER

Arguments

Name	Type	Description
filename	string	Filename you want to read
filetype (optional)	constant	Filetype you want to read. Can be Model.LSDYNA, Model.ABAQUS, Model.NASTRAN, Model.RADIOSS or Model.IGES. If omitted the file is assumed to be a DYNA3D file. For Model.NASTRAN there are options that change how the model is read. See Options for details.
number (optional)	integer	Model number to read file into. If omitted the next free model number will be used.

Return type

Model object (or null if error)

Example

To read the keyword file /data/test/file.key

```
Model.Read("/data/test/file.key");
```

To read the NASTRAN file /data/test/file.dat

```
Model.Read("/data/test/file.dat", Model.NASTRAN);
```

To read the keyword file /data/test/file.key into model 10

```
Model.Read("/data/test/file.key", Model.LSDYNA, 10);
```

RenumberAll(start*[integer]*)

Description

Renumbers all of the items in the model.

Arguments

Name	Type	Description
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber everything in model m, starting at 1000000:

```
m.RenumberAll(1000000);
```

RenumberFlagged(flag*[Flag]*, start*[integer]*, mode (optional)*[constant]*)

Description

Renumbers all of the flagged items in the model.

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to renumber
start	integer	Start point for renumbering
mode (optional)	constant	Renumber mode. Can be Model.IGNORE_CLASH, Model.MOVE_CLASH_UP, Model.SHIFT_ALL_UP, or Model.RENUMBER_TO_FREE(Default),

Return type

No return value

Example

To renumber everything in model m flagged with flag f, starting at 1000000, using mode MOVE_CLASH_UP:

```
m.RenumberFlagged(f, 1000000, Model.MOVE_CLASH_UP);
```

Select(prompt*[string]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select a model using standard PRIMER object menus. If there are no models in memory then Select returns null. If only one model is present then the model object is returned. If there is more than one model in memory then an object menu is mapped allowing the user to choose a model.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Model object (or null if no models present).

Example

To select a model giving the prompt 'Select model':

```
var m = Model.Select('Select model');
```

SetColour(colour*[colour from Colour class.]*)

Description

Sets the colour of the model.

Arguments

Name	Type	Description
colour	colour from Colour class.	The colour you want to set the model to

Return type

No return value

Example

To set the colour of model m to red:

```
m.SetColour(Colour.RED);
```

or

```
m.SetColour(Colour.RGB(255, 0, 0));
```

SetFlag(flag*[Flag]*)

Description

Sets the flagging for a model in PRIMER. See also Model.PropagateFlag(), Model.ClearFlag(), global.AllocateFlag() and global.ReturnFlag().

Arguments

Name	Type	Description
flag	Flag	Flag to set

Return type

No return value

Example

To set flag f for everything in model m:

```
m.SetFlag(f);
```

Show()

Description

Shows a model in PRIMER

Arguments

No arguments

Return type

No return value

Example

To show model m in PRIMER

```
m.Show();
```

Total() [static]

Description

Returns the total number of models.

Arguments

No arguments

Return type

integer

Example

To find how many models there are in PRIMER:

```
var num = Model.Total();
```

Unblank()

Description

Unblanks a model in PRIMER

Arguments

No arguments

Return type

No return value

Example

To unblank model m

```
m.Unblank();
```

UnblankAll() [static]

Description

Unblanks all models

Arguments

No arguments

Return type

No return value

Example

To unblank all models

```
Model.UnblankAll();
```

UnblankFlagged(flag*[Flag]*, redraw (optional)*[boolean]*)

Description

Unblanks all of the flagged items in the model.

Arguments

Name	Type	Description
flag	Flag	Flag set on items that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank everything in model m flagged with flag f:

```
m.UnblankFlagged(f);
```

UnsketchAll(redraw (optional)*[boolean]*)

Description

Unsketches all of the sketched items in the model.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the items are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all the sketched objects in model m:

```
m.UnsketchAll();
```

UpdateGraphics()

Description

Updates the graphics for a model in PRIMER

Arguments

No arguments

Return type

No return value

Example

To update the graphics for model m

```
m.UpdateGraphics();
```

UsesLargeLabels()

Description

Checks to see if a model uses large labels

Arguments

No arguments

Return type

true if model uses large labels, false otherwise

Example

To check if model m uses large labels:

```
var large = m.UsesLargeLabels();
```

Write(filename*[string]*, options (optional)*[object]*)

Description

Writes a model in PRIMER to file

Arguments

Name	Type	Description
filename	string	Filename of the LS-DYNA keyword file you want to write

<table>
<tr><td>options (optional)</td><td>object</td><td>Options specifying how the file should be written out. If omitted the default values below will be used. The properties available are: Object has the following properties:
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>binary (optional)</td><td>boolean</td><td>If true then the entire output file will be written out in binary, if false then the entire file will be written in ascii. If not defined the default is for each file in the model to be written in its original format, or if this is a newly created model ascii will be used.</td></tr>
<tr><td>compress (optional)</td><td>boolean</td><td>If true then the output file will be compressed. If false (default) then an uncompressed file will be written.</td></tr>
<tr><td>compressLevel (optional)</td><td>integer</td><td>Compression level for .gz and .zip files. Must be in the range 1 to 9 with 1 being the least compression (fastest speed) to 9 being the greatest compression (slowest speed)</td></tr>
<tr><td>compressMode (optional)</td><td>integer</td><td>This option can be used to specify the mode of compression. Can be Include.KEEP_ORIGINAL or Include.INDIVIDUAL_GZIP or Include.INDIVIDUAL_ZIP</td></tr>
<tr><td>fileStartAscii (optional)</td><td>boolean</td><td>Only relevant if binary output format has been selected. If true then the beginning of the file (*CONTROL etc) file is written out in ascii. If false (default) then the entire file is converted to binary.</td></tr>
<tr><td>i10 (optional)</td><td>boolean</td><td>If true then i10 format will be used to write the file. If false (default) then the original LS-DYNA format in which the file was written will be used, or if this is a newly created model the smallest format which will contain it will be used. Note that large format is only available from version R9 and above. See also the large property.</td></tr>
<tr><td>kbyExt (optional)</td><td>boolean</td><td>If true then a binary format output file will be given the extension .kby, replacing the existing extension.</td></tr>
<tr><td>large (optional)</td><td>boolean</td><td>If true then large format will be used to write the file. If false (default) then the normal LS-DYNA format will be used. Note that large format is only available from version R7.1 and above.</td></tr>
<tr><td>masterAscii (optional)</td><td>boolean</td><td>Only relevant if binary output format has been selected. If true then the whole master file is written out in ascii. If false (default) then the master file is also converted to binary.</td></tr>
<tr><td>method (optional)</td><td>integer</td><td>The method used to write include files. Can be Include.MASTER_ONLY, Include.MERGE, Include.SELECT, Include.NOT_WRITTEN, Include.SUBDIR (default) or Include.SAME_DIR</td></tr>
<tr><td>parametersAsValues (optional)</td><td>boolean</td><td>If true then the underlying values of any parameters will be written when they are used in data fields rather than ’&name’. If false then ’&name’ will be written. See also Options.keyout_parameter_values</td></tr>
<tr><td>path (optional)</td><td>integer</td><td>The method used to write include paths. Can be Include.ABSOLUTE (default) or Include.RELATIVE</td></tr>
<tr><td>separator (optional)</td><td>integer</td><td>The directory separator used when writing include files. Can be Include.NATIVE (default), Include.UNIX or Include.WINDOWS</td></tr>
<tr><td>version (optional)</td><td>string</td><td>The LS-DYNA version used to write the file. Can be "971R5", "971R4", "971R3", "970v6763" etc (see the version popup in Model->Write ’>>> LS-DYNA output options’ for a full list). See also Options.dyna_version</td></tr>
</table>
</td></tr>
</table>

Return type

No return value

Example

To Write model m to file /data/test/file.key as a compressed gzip in version R10.0

```
var output_obj = new Object();
output_obj.version = "R10.0";
output_obj.compress = true;
output_obj.compressMode = Model.INDIVIDUAL_GZIP;
m.Write("/data/test/file.key", output_obj);
```

Write(filename*[string]*, method (optional)*[constant]*, path (optional)*[constant]*, separator (optional)*[constant]*, version (optional)*[string]*, large (optional)*[boolean]*) [deprecated]

This function is deprecated in version 15.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Writes a model in PRIMER to file

Arguments

Name	Type	Description
filename	string	Filename of the LS-DYNA keyword file you want to write
method (optional)	constant	The method used to write include files. Can be Include.MASTER_ONLY, Include.MERGE, Include.SELECT, Include.NOT_WRITTEN, Include.SUBDIR (default) or Include.SAME_DIR
path (optional)	constant	The method used to write include paths. Can be Include.ABSOLUTE (default) or Include.RELATIVE
separator (optional)	constant	The directory separator used when writing include files. Can be Include.NATIVE (default), Include.UNIX or Include.WINDOWS
version (optional)	string	The LS-DYNA version used to write the file. Can be "971R5", "971R4", "971R3", "970v6763" etc (see the version popup in Model->Write '>>> LS-DYNA output options' for a full list). See also Options.dyna_version
large (optional)	boolean	If true then large format will be used to write the file. If false (default) then the original LS-DYNA format in which the file was written will be used, or if this is a newly created model the smallest format which will contain it will be used. Note that large format is only available from version R7.1 and above. See also the i10 property.

Return type

No return value

Example

To Write model m to file /data/test/file.key

```
m.Write("/data/test/file.key");
```

MorphBox class

The MorphBox class gives you access to morph boxes in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- BlankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- First(Model *[Model]*)
- FirstFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- FlagAll(Model *[Model]*, flag *[Flag]*)
- FlagAllMorphedConnections(model *[Model]*, flag *[integer]*)
- ForEach(Model *[Model]*, func *[function]*, extra (optional) *[any]*)
- GetAll(Model *[Model]*)
- GetFlagged(Model *[Model]*, flag *[Flag]*)
- GetFromID(Model *[Model]*, number *[integer]*)
- Last(Model *[Model]*)
- LastFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- NextFreeLabel(Model *[Model]*, layer (optional) *[Include number]*)
- Pick(prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*, button text (optional) *[string]*)
- RenumberAll(Model *[Model]*, start *[integer]*)
- RenumberFlagged(Model *[Model]*, flag *[Flag]*, start *[integer]*)
- Select(flag *[Flag]*, prompt *[string]*, limit (optional) *[Model or Flag]*, modal (optional) *[boolean]*)
- SetMorphConnections(status *[boolean]*)
- SketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- Total(Model *[Model]*, exists (optional) *[boolean]*)
- UnblankAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnblankFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)
- UnflagAll(Model *[Model]*, flag *[Flag]*)
- UnsketchAll(Model *[Model]*, redraw (optional) *[boolean]*)
- UnsketchFlagged(Model *[Model]*, flag *[Flag]*, redraw (optional) *[boolean]*)

Member functions

- ApplyMorphing(redraw (optional) *[boolean]*)
- AssociateComment(Comment *[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag *[Flag]*)
- Copy(range (optional) *[boolean]*)
- DetachComment(Comment *[Comment]*)
- Error(message *[string]*, details (optional) *[string]*)
- FlagMorphedConnections(flag *[integer]*)
- Flagged(flag *[Flag]*)
- GetComments()
- GetParameter(prop *[string]*)
- GetPoint(xindex *[integer]*, yindex *[integer]*, zindex *[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- Reset(redraw (optional) *[boolean]*)
- SetFlag(flag *[Flag]*)
- SetPointID(xindex *[integer]*, yindex *[integer]*, zindex *[integer]*, id *[integer]*)
- Sketch(redraw (optional) *[boolean]*)
- Unblank()
- Unsketch(redraw (optional) *[boolean]*)

- UpdateParametricCoordinates()
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

MorphBox properties

Name	Type	Description
exists	logical	true if morph box exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the morph box is in.
label	integer	MorphBox number.
model	integer	The Model number that the box is in.
nx	integer	Number of morph points in parametric X direction (read only)
ny	integer	Number of morph points in parametric Y direction (read only)
nz	integer	Number of morph points in parametric Z direction (read only)
setid	integer	ID for node set of nodes dragged with the box (read only). This will be a *SET_NODE_COLUMN containing the nodes together with their parametric coordinates in X, Y, Z. It is strongly discouraged to edit the contents of this set or the column data manually.

Detailed Description

The MorphBox class allows you to create, modify and manipulate morph boxes. See the documentation below for more details.

Constructor

new MorphBox(Model*[Model]*, label*[integer]*, flag*[Flag]*, options (optional)*[object]*)

Description

Create a new MorphBox object around flagged items.

Arguments

Name	Type	Description
Model	Model	Model that morph box will be created in
label	integer	MorphBox number
flag	Flag	Flag set on the entities (for example nodes, elements and/or parts) that you want to create the box around

<table>
<tr><td>options (optional)</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>csys (optional)</td><td>integer</td><td>Coordinate system for local coordinates. Leave undefined if using global coordinates or if local coordinate system defined with n1, n2 and n3.</td></tr>
<tr><td>n1 (optional)</td><td>integer</td><td>Node 1 label for local coordinate. Leave undefined if using global coordinates or if local coordinate system defined with csys.</td></tr>
<tr><td>n2 (optional)</td><td>integer</td><td>Node 2 label for local coordinate. Leave undefined if using global coordinates or if local coordinate system defined with csys.</td></tr>
<tr><td>n3 (optional)</td><td>integer</td><td>Node 3 label for local coordinate. Leave undefined if using global coordinates or if local coordinate system defined with csys.</td></tr>
<tr><td>nx (optional)</td><td>integer</td><td>Number of points in X direction of box (assumed to be 2 for linear box if omitted)</td></tr>
<tr><td>ny (optional)</td><td>integer</td><td>Number of points in Y direction of box (assumed to be 2 for linear box if omitted)</td></tr>
<tr><td>nz (optional)</td><td>integer</td><td>Number of points in Z direction of box (assumed to be 2 for linear box if omitted)</td></tr>
<tr><td>points (optional)</td><td>Array of integers</td><td>Array of integers of depth 3 containing the morph point IDs. This should be omitted in the (default) case of also creating new morph points together with the morph box at the locations based on the bounding box of the flagged items. If this array contains ’nx’ by ’ny’ by ’nz’ existing morph points, the morph box is attached to these points, and ’csys’, ’n1’, ’n2’, ’n3’ will be irrelevant. Each ’points[i][j][k]’ should contain the morph point ID to be added at index i in local X direction, index j in local Y direction and index k in local Z direction. The box will then still contain flagged nodes only, but nodes geometrically outside the volume of the morph points will not be included either.</td></tr>
</table>
Options to create the box. Object has the following properties:
</td></tr>
</table>

Return type

MorphBox object

Example

To create a new morph box in model m with label 100 and 2 by 2 by 2 points (linear in each coordinate direction) around all items flagged with flag in global coordinates:

```
var box = new MorphBox(m, 100, flag);
```

To create a new morph box in model m with label 100 and 4 by 4 by 2 points (cubic in parametric X and Y directions and linear in Z direction) around all flagged items in local coordinates determined by nodes 11, 12 and 13:

```
var options = new Object();
options.nx = 4;
options.ny = 4;
options.nz = 2;
options.n1 = 11;
options.n2 = 12;
options.n3 = 13;
var box = new MorphBox(m, 100, flag, options);
```

Suppose there are already morph points 1, 2, 3, 4, 5, 6, 7, 8 in model m at coordinates (0, 0, 0), (0, 0, 100), (0, 100, 0), (0, 100, 100), (100, 0, 0), (100, 0, 100), (100, 100, 0), (100, 100, 100) respectively. To create a new linear morph box between these points containing flagged items inside their volume:

```
var options = new Object();
options.points = [[[1,2],[3,4]],[[5,6],[7,8]]];
var box = new MorphBox(m, 100, flag, options);
```

Details of functions

ApplyMorphing(redraw (optional)*[boolean]*)

Description

Recalculates the X, Y and Z coordinates of all nodes linked to the morph box by the *SET_NODE_COLUMN. This should be called when coordinates of morph points have changed and you wish to apply the morphing. If several morph point positions on the same box change, then it is more speed-efficient to call this function only once for the box.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to apply the morphing to several boxes and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using Model.UpdateGraphics().

Return type

No return value

Example

To calculate all global X, Y and Z coordinates for the morphed nodes for box b:

```
b.ApplyMorphing();
```

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a box.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the box

Return type

No return value

Example

To associate comment c to the box b:

```
b.AssociateComment(c);
```

Blank()

Description

Blanks the box

Arguments

No arguments

Return type

No return value

Example

To blank box b:

```
b.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boxs in model m:

```
MorphBox.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be blanked in
flag	Flag	Flag set on the boxs that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the boxs in model m flagged with f:

```
MorphBox.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the box is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if box b is blanked:

```
if (b.Blanked() ) do_something...
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the box.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the box

Return type

No return value

Example

To clear flag f for box b:

```
b.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the box.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

MorphBox object

Example

To copy box b into box z:

```
var z = b.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a box.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the box

Return type

No return value

Example

To detach comment c from the box b:

```
b.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for box. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for box b:

```
b.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first box in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first box in

Return type

MorphBox object (or null if there are no boxs in the model).

Example

To get the first box in model m:

```
var b = MorphBox.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free box label in the model. Also see MorphBox.LastFreeLabel(), MorphBox.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free box label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

MorphBox label.

Example

To get the first free box label in model m:

```
var label = MorphBox.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the boxs in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be flagged in
flag	Flag	Flag to set on the boxs

Return type

No return value

Example

To flag all of the boxs with flag f in model m:

```
MorphBox.FlagAll(m, f);
```

FlagAllMorphedConnections(model*[Model]*, flag*[integer]*) [static]

Description

Flags all connections, in a given model, that have been morphed since their last remake. This includes connections that have been morphed by a morph box that has since been deleted.

Arguments

Name	Type	Description
model	Model	Model containing desired connections.
flag	integer	Flag to mark morphed connections.

Return type

true if successful, false if not.

Example

To flag all morphed connections in Model m with flag.

```
var flag = AllocateFlag();
   MorphBox.FlagAllMorphedConnections(m, flag);
```

FlagMorphedConnections(flag*[integer]*)

Description

Flags all connections that have been morphed, by a givine morph box, since their last remake. A connection could be morphed by one morph box and not another, therefore calling this function on two boxes that share a connection may produce different results depending on which box the function is called for. E.g. morb1 and morb2 share conx1, morb1 gets morphed whereas morb2 remains unchanged. Calling this function for morb1 will flag conx1, however calling the function for morb2 won't flag conx1.

Arguments

Name	Type	Description
flag	integer	Flag to mark morphed connections.

Return type

true if successful, false if not.

Example

To flag all morphed connections in a MorphBox with flag.

```
var flag = AllocateFlag();
   box.FlagMorphedConnections(flag);
```

Flagged(flag*[Flag]*)

Description

Checks if the box is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the box

Return type

true if flagged, false if not.

Example

To check if box b has flag f set on it:

```
if (b.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each box in the model.
Note that ForEach has been designed to make looping over boxs as fast as possible and so has some limitations. Firstly, a single temporary MorphBox object is created and on each function call it is updated with the current box data. This means that you should not try to store the MorphBox object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new boxs inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all boxs are in
func	function	Function to call for each box
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the boxs in model m:

```
MorphBox.ForEach(m, test);
function test(b)
{
// b is MorphBox object
}
```

To call function test for all of the boxs in model m with optional object:

```
var data = { x:0, y:0 };
MorphBox.ForEach(m, test, data);
function test(b, extra)
{
// b is MorphBox object
// extra is data
}
```

GetAll(Model*[Model]*) [static]

Description

Returns an array of MorphBox objects for all of the boxs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boxs from

Return type

Array of MorphBox objects

Example

To make an array of MorphBox objects for all of the boxs in model m

```
var b = MorphBox.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a box.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the box b:

```
var comm_array = b.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of MorphBox objects for all of the flagged boxs in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get boxs from
flag	Flag	Flag set on the boxs that you want to retrieve

Return type

Array of MorphBox objects

Example

To make an array of MorphBox objects for all of the boxs in model m flagged with f

```
var b = MorphBox.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the MorphBox object for a box ID.

Arguments

Name	Type	Description
Model	Model	Model to find the box in
number	integer	number of the box you want the MorphBox object for

Return type

MorphBox object (or null if box does not exist).

Example

To get the MorphBox object for box 100 in model m

```
var b = MorphBox.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a MorphBox property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the MorphBox.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	box property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if MorphBox property b.example is a parameter:

```
Options.property_parameter_names = true;
if (b.GetParameter(b.example) ) do_something...
Options.property_parameter_names = false;
```

To check if MorphBox property b.example is a parameter by using the GetParameter method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

GetPoint(xindex*[integer]*, yindex*[integer]*, zindex*[integer]*)

Description

Returns the morph point ID on the morph box at indices in X, Y and Z directions.

Arguments

Name	Type	Description
xindex	integer	Index of the point in X direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric X coordinate and box.nx-1 for the points with the highest X.
yindex	integer	Index of the point in Y direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric Y coordinate and box.ny-1 for the points with the highest Y.
zindex	integer	Index of the point in Z direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric Z coordinate and box.nz-1 for the points with the highest Z.

Return type

A MorphPoint object for the point on the box at given indices.

Example

To get the 2nd point on the edge along the local Y direction and at highest local X and lowest local Z coordinate:

```
var point = box.GetPoint(box.nx-1, 1, 0);
```

Keyword()

Description

Returns the keyword for this morph box (*MORPH_BOX or *MORPH_BOX_HIGH_ORDER). **Note that a carriage return is not added**. See also MorphBox.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for morph box b:

```
var key = b.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the morph box. **Note that a carriage return is not added**. See also MorphBox.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for morph box b:

```
var cards = b.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last box in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last box in

Return type

MorphBox object (or null if there are no boxs in the model).

Example

To get the last box in model m:

```
var b = MorphBox.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free box label in the model. Also see MorphBox.FirstFreeLabel(), MorphBox.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free box label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

MorphBox label.

Example

To get the last free box label in model m:

```
var label = MorphBox.LastFreeLabel(m);
```

Next()

Description

Returns the next box in the model.

Arguments

No arguments

Return type

MorphBox object (or null if there are no more boxs in the model).

Example

To get the box in model m after box b:

```
var b = b.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) box label in the model. Also see MorphBox.FirstFreeLabel(), MorphBox.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free box label in

layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

MorphBox label.

Example

To get the next free box label in model m:

```
var label = MorphBox.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a box.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boxs from that model can be picked. If the argument is a Flag then only boxs that are flagged with *limit* can be selected. If omitted, or null, any boxs from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

MorphBox object (or null if not picked)

Example

To pick a box from model m giving the prompt 'Pick box from screen':

```
var b = MorphBox.Pick('Pick box from screen', m);
```

Previous()

Description

Returns the previous box in the model.

Arguments

No arguments

Return type

MorphBox object (or null if there are no more boxs in the model).

Example

To get the box in model m before box b:

```
var b = b.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boxs in model m, from 1000000:

```
MorphBox.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be renumbered in
flag	Flag	Flag set on the boxs that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the boxs in model m flagged with f, from 1000000:

```
MorphBox.RenumberFlagged(m, f, 1000000);
```

Reset(redraw (optional)*[boolean]*)

Description

Resets the morph box to its initial position and updates the coordinates of all its nodes.

Arguments

Name	Type	Description

redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to reset several boxes and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using Model.UpdateGraphics().

Return type

No return value

Example

To reset box b:

```
b.Reset();
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select boxs using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting boxs
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only boxs from that model can be selected. If the argument is a Flag then only boxs that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any boxs can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of boxs selected or null if menu cancelled

Example

To select boxs from model m, flagging those selected with flag f, giving the prompt 'Select boxs':

```
MorphBox.Select(f, 'Select boxs', m);
```

To select boxs, flagging those selected with flag f but limiting selection to boxs flagged with flag l, giving the prompt 'Select boxs':

```
MorphBox.Select(f, 'Select boxs', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the box.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the box

Return type

No return value

Example

To set flag f for box b:

```
b.SetFlag(f);
```

SetMorphConnections(status*[boolean]*) [static]

Description

Turns Morph Connections on/off.

Arguments

Name	Type	Description
status	boolean	true turns Morph Connections on. false turns Morph Connections off.

Return type

No return value.

Example

To turn Morph Connections on.

```
MorphBox.SetMorphConnections(true);
```

SetPointID(xindex*[integer]*, yindex*[integer]*, zindex*[integer]*, id*[integer]*)

Description

Replaces the morph point ID on the array, whose size depends on the orders in X, Y and Z directions, with the given new ID.

Arguments

Name	Type	Description
xindex	integer	Index of the point in X direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric X coordinate and box.nx-1 for the points with the highest X.
yindex	integer	Index of the point in Y direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric Y coordinate and box.ny-1 for the points with the highest Y.
zindex	integer	Index of the point in Z direction. Note that indices start at 0, so it should be 0 for the points with the smallest parameteric Z coordinate and box.nz-1 for the points with the highest Z.
id	integer	New MorphPoint id.

Return type

No return value

Example

To replace the 2nd point on the edge along the local X direction and at lowest local Y and highest local Z coordinate with point 101:

```
box.SetPointID(1, 0, box.nz-1, 101);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the box. The box will be sketched until you either call MorphBox.Unsketch(), MorphBox.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the box is sketched. If omitted redraw is true. If you want to sketch several boxs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch box b:

```
b.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged boxs in the model. The boxs will be sketched until you either call MorphBox.Unsketch(), MorphBox.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged boxs will be sketched in
flag	Flag	Flag set on the boxs that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the boxs are sketched. If omitted redraw is true. If you want to sketch flagged boxs several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all boxs flagged with flag in model m:

```
MorphBox.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for

exists (optional)	boolean	true if only existing boxs should be counted. If false or omitted referenced but undefined boxs will also be included in the total.

Return type

number of boxs

Example

To get the total number of boxs in model m:

```
var total = MorphBox.Total(m);
```

Unblank()

Description

Unblanks the box

Arguments

No arguments

Return type

No return value

Example

To unblank box b:

```
b.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boxs in model m:

```
MorphBox.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged boxs will be unblanked in
flag	Flag	Flag set on the boxs that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the boxs in model m flagged with f:

```
MorphBox.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all boxs will be unset in
flag	Flag	Flag to unset on the boxs

Return type

No return value

Example

To unset the flag f on all the boxs in model m:

```
MorphBox.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the box.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the box is unsketched. If omitted redraw is true. If you want to unsketch several boxs and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch box b:

```
b.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all boxs.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the boxs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boxs in model m:

```
MorphBox.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged boxs in the model.

Arguments

Name	Type	Description
Model	Model	Model that all boxs will be unsketched in
flag	Flag	Flag set on the boxs that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the boxs are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all boxs flagged with flag in model m:

```
MorphBox.UnsketchAll(m, flag);
```

UpdateParametricCoordinates()

Description

Recalculates parametric X, Y, Z coordinates for each node in the *SET_NODE_COLUMN associated with the morph box. This needs to be called whenever morph points on the box or their coordinates have been changed manually and you wish to keep all nodes at their intrinsic global X, Y, Z coordinates. Provided Morph Connections is on (see MorphBox.SetMorphConnections()), this will also force PRIMER to recalculate the parametric coordinates for any connections in the morph box next time one of its morph points is moved.

Arguments

No arguments

Return type

No return value

Example

To recalculate all X, Y and Z coordinates for box b:

```
b.UpdateParametricCoordinates();
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

MorphBox object.

Example

To check if MorphBox property b.example is a parameter by using the MorphBox.GetParameter() method:

```
if (b.ViewParameters().GetParameter(b.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for box. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for box b:

```
b.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this box.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for box b:

```
var xrefs = b.Xrefs();
```

---

toString()

Description

Creates a string containing the morph box data in keyword format. Note that this contains the keyword header and the keyword cards. See also MorphBox.Keyword() and MorphBox.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for morph box b in keyword format

```
var s = b.toString();
```

MorphFlow class

The MorphFlow class gives you access to morph flows in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Create(Model*[Model]*, modal (optional)*[boolean]*)
- First(Model*[Model]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- GetFromName(Model*[Model]*, morph flow name*[string]*)
- Last(Model*[Model]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- Browse(modal (optional)*[boolean]*)
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Edit(modal (optional)*[boolean]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- GetRow(row*[integer]*)
- GetValue(index*[integer]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- RemoveRow(row*[integer]*)
- RemoveValue(index*[integer]*)
- SetFlag(flag*[Flag]*)
- SetRow(row*[integer]*, data*[Array of data]*)
- SetValue(index*[integer]*, value*[real]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)

- Xrefs()
- toString()

MorphFlow properties

Name	Type	Description
exists	logical	true if morph flow exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the morph flow is in.
max	real	Maximum value for type set to "CONTINUOUS" or "STEP" when written as design variable for LS-OPT.
min	real	Minimum value for type set to "CONTINUOUS" or "STEP" when written as design variable for LS-OPT.
model	integer	The Model number that the flow is in.
name	string	Name of the morph flow. If the flow is used for applying LS-OPT variables, this should match the variable name in the listing file written by LS-OPT.
npoints	integer	Number of morph points referenced by the flow. (read only)
nvals	integer	Number of values in the list when type is set to "DISCRETE". (read only)
step	real	Step size for type set to "STEP" when written as design variable for LS-OPT.
type	string	Range type for the morph flow. This should be "CONTINUOUS", "STEP" or "DISCRETE" and may be used for LS-OPT when writing design variable files from morph flows.

Detailed Description

The MorphFlow class allows you to create, modify and manipulate morph flows. See the documentation below for more details.

Constructor

new MorphFlow(Model*[Model]*, name*[string]*)

Description

Create a new MorphFlow object.

Arguments

Name	Type	Description
Model	Model	Model that morph flow will be created in
name	string	MorphFlow name

Return type

MorphFlow object

Example

To create a new (empty) morph flow in model m with name ’depth’

```
var f = new MorphFlow(m, "depth");
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a flow.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the flow

Return type

No return value

Example

To associate comment c to the flow flow:

```
flow.AssociateComment(c);
```

Blank()

Description

Blanks the flow

Arguments

No arguments

Return type

No return value

Example

To blank flow flow:

```
flow.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that all flows will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the flows in model m:

```
MorphFlow.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged flows will be blanked in
flag	Flag	Flag set on the flows that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the flows in model m flagged with f:

```
MorphFlow.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the flow is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if flow flow is blanked:

```
if (flow.Blanked() ) do_something...
```

Browse(modal (optional)*[boolean]*)

Description

Starts an edit panel in Browse mode.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Browse flow flow:

```
flow.Browse();
```

ClearFlag(flag*[Flag]*)

Description

Clears a flag on the flow.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the flow

Return type

No return value

Example

To clear flag f for flow flow:

```
flow.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the flow.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

MorphFlow object

Example

To copy flow flow into flow z:

```
var z = flow.Copy();
```

Create(Model*[Model]*, modal (optional)*[boolean]*) [static]

Description

Starts an interactive editing panel to create a morph flow card.

Arguments

Name	Type	Description
Model	Model	Model that the morph flow card will be created in
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

MorphFlow object (or null if not made)

Example

To start creating a morph flow card in model m:

```
var f = MorphFlow.Create(m);
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a flow.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the flow

Return type

No return value

Example

To detach comment c from the flow flow:

```
flow.DetachComment(c);
```

Edit(modal (optional)*[boolean]*)

Description

Starts an interactive editing panel.

Arguments

Name	Type	Description
modal (optional)	boolean	If this window is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the window will be modal.

Return type

no return value

Example

To Edit flow flow:

```
flow.Edit();
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for flow. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for flow flow:

```
flow.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first flow in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first flow in

Return type

MorphFlow object (or null if there are no flows in the model).

Example

To get the first flow in model m:

```
var flow = MorphFlow.First(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the flows in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all flows will be flagged in
flag	Flag	Flag to set on the flows

Return type

No return value

Example

To flag all of the flows with flag f in model m:

```
MorphFlow.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the flow is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the flow

Return type

true if flagged, false if not.

Example

To check if flow flow has flag f set on it:

```
if (flow.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each flow in the model.
Note that ForEach has been designed to make looping over flows as fast as possible and so has some limitations. Firstly, a single temporary MorphFlow object is created and on each function call it is updated with the current flow data. This means that you should not try to store the MorphFlow object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new flows inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all flows are in
func	function	Function to call for each flow
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the flows in model m:

```
MorphFlow.ForEach(m, test);
function test(flow)
{
// flow is MorphFlow object
}
```

To call function test for all of the flows in model m with optional object:

```
var data = { x:0, y:0 };
MorphFlow.ForEach(m, test, data);
function test(flow, extra)
{
// flow is MorphFlow object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of MorphFlow objects for all of the flows in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get flows from

Return type

Array of MorphFlow objects

Example

To make an array of MorphFlow objects for all of the flows in model m

```
var flow = MorphFlow.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a flow.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the flow flow:

```
var comm_array = flow.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of MorphFlow objects for all of the flagged flows in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get flows from
flag	Flag	Flag set on the flows that you want to retrieve

Return type

Array of MorphFlow objects

Example

To make an array of MorphFlow objects for all of the flows in model m flagged with f

```
var flow = MorphFlow.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the MorphFlow object for a flow ID.

Arguments

Name	Type	Description
Model	Model	Model to find the flow in
number	integer	number of the flow you want the MorphFlow object for

Return type

MorphFlow object (or null if flow does not exist).

Example

To get the MorphFlow object for flow 100 in model m

```
var flow = MorphFlow.GetFromID(m, 100);
```

GetFromName(Model*[Model]*, morph flow name*[string]*) [static]

Description

Returns the stored MorphFlow object for a morph flow name. WARNING: This assumes that there is at most one morph flow with a given name. Otherwise this function only returns the first occurrence.

Arguments

Name	Type	Description
Model	Model	Model to find the morph flow in
morph flow name	string	name of the morph flow you want the MorphFlow object for

Return type

MorphFlow object (or null if morph flow does not exist).

Example

To get the MorphFlow object for flow "depth" in model m

```
var f = MorphFlow.GetFromName(m, "depth");
```

GetParameter(prop*[string]*)

Description

Checks if a MorphFlow property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the MorphFlow.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	flow property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if MorphFlow property flow.example is a parameter:

```
Options.property_parameter_names = true;
if (flow.GetParameter(flow.example) ) do_something...
Options.property_parameter_names = false;
```

To check if MorphFlow property flow.example is a parameter by using the GetParameter method:

```
if (flow.ViewParameters().GetParameter(flow.example) ) do_something...
```

GetRow(row*[integer]*)

Description

Returns the data for a row in the morph flow.

Arguments

Name	Type	Description
row	integer	The row you want the data for. **Note row indices start at 0**.

Return type

An array of numbers containing the morph point ID at index 0 and the vector components at indices 1, 2, 3.

Example

To get the data for the 2nd row in morph flow f:

```
var data = f.GetRow(1);
var point_id = data[0];
var dx = data[1];
var dy = data[2];
var dz = data[3];
```

GetValue(index*[integer]*)

Description

Get the value at given index on the morph flow with type "DISCRETE".

Arguments

Name	Type	Description
index	integer	The index where you are extracting the value. **Note row indices start at 0**.

Return type

real

Example

To get the 2nd value for morph flow f with type "DISCRETE":

```
var value = f.GetValue(1);
```

To get the last value on the list of values on f:

```
var value = f.GetValue(f.nvals - 1);
```

Keyword()

Description

Returns the keyword for this morph flow (*MORPH_FLOW). **Note that a carriage return is not added**. See also MorphFlow.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for morph flow f:

```
var key = f.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the morph flow. **Note that a carriage return is not added**. See also MorphFlow.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for morph flow f:

```
var cards = f.KeywordCards();
```

Last(Model*[Model]*) [static]

Description

Returns the last flow in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last flow in

Return type

MorphFlow object (or null if there are no flows in the model).

Example

To get the last flow in model m:

```
var flow = MorphFlow.Last(m);
```

Next()

Description

Returns the next flow in the model.

Arguments

No arguments

Return type

MorphFlow object (or null if there are no more flows in the model).

Example

To get the flow in model m after flow flow:

```
var flow = flow.Next();
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a flow.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only flows from that model can be picked. If the argument is a Flag then only flows that are flagged with *limit* can be selected. If omitted, or null, any flows from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

MorphFlow object (or null if not picked)

Example

To pick a flow from model m giving the prompt 'Pick flow from screen':

```
var flow = MorphFlow.Pick('Pick flow from screen', m);
```

Previous()

Description

Returns the previous flow in the model.

Arguments

No arguments

Return type

MorphFlow object (or null if there are no more flows in the model).

Example

To get the flow in model m before flow flow:

```
var flow = flow.Previous();
```

RemoveRow(row*[integer]*)

Description

Removes the data (a morph point ID and its three vector components) for a row in *MORPH_FLOW.

Arguments

Name	Type	Description
row	integer	The row you want to remove the data for. **Note that row indices start at 0**.

Return type

No return value.

Example

To remove the second row of data for morph flow f:

```
f.RemoveRow(1);
```

RemoveValue(index*[integer]*)

Description

Removes the value at given index in *MORPH_FLOW with type "DISCRETE".

Arguments

Name	Type	Description
index	integer	The index where you are removing the value. **Note that indices start at 0**.

Return type

No return value.

Example

To remove the second value for morph flow f:

```
f.RemoveValue(1);
```

To remove the last value for f:

```
f.RemoveValue(f.nvals - 1);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select flows using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting flows
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only flows from that model can be selected. If the argument is a Flag then only flows that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any flows can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of flows selected or null if menu cancelled

Example

To select flows from model m, flagging those selected with flag f, giving the prompt ’Select flows’:

```
MorphFlow.Select(f, ’Select flows’, m);
```

To select flows, flagging those selected with flag f but limiting selection to flows flagged with flag l, giving the prompt ’Select flows’:

```
MorphFlow.Select(f, ’Select flows’, l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the flow.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the flow

Return type

No return value

Example

To set flag f for flow flow:

```
flow.SetFlag(f);
```

SetRow(row*[integer]*, data*[Array of data]*)

Description

Sets the data for a row in *MORPH_FLOW.

Arguments

Name	Type	Description
row	integer	The row you want to set the data for. **Note that row indices start at 0**.
data	Array of data	The data you want to set the row to. It should be of length 4 having the morph point ID at index 0, and the vector components at indices 1, 2, 3.

Return type

No return value.

Example

To set the second point of the morph flow f to be morph point 11 with unit vector in X-direction:

```
var array = [11, 1.0, 0.0, 0.0];
f.SetRow(1, array);
```

To append a new row of data (using the same array of values):

```
f.SetRow(f.npoints, array);
```

SetValue(index*[integer]*, value*[real]*)

Description

Sets the value at given index in a *MORPH_FLOW with type "DISCRETE".

Arguments

Name	Type	Description
index	integer	The row you want to set the data for. **Note that row indices start at 0**.
value	real	The new value to insert into the list.

Return type

No return value.

Example

To set the second value morph flow f to 20.0:

```
f.SetValue(1, 20.0);
```

To append the value 20.0 to the end of the list:

```
f.SetValue(f.nvals, 20.0);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the flow. The flow will be sketched until you either call MorphFlow.Unsketch(), MorphFlow.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the flow is sketched. If omitted redraw is true. If you want to sketch several flows and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch flow flow:

```
flow.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged flows in the model. The flows will be sketched until you either call MorphFlow.Unsketch(), MorphFlow.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged flows will be sketched in
flag	Flag	Flag set on the flows that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the flows are sketched. If omitted redraw is true. If you want to sketch flagged flows several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all flows flagged with flag in model m:

```
MorphFlow.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of flows in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing flows should be counted. If false or omitted referenced but undefined flows will also be included in the total.

Return type

number of flows

Example

To get the total number of flows in model m:

```
var total = MorphFlow.Total(m);
```

Unblank()

Description

Unblanks the flow

Arguments

No arguments

Return type

No return value

Example

To unblank flow flow:

```
flow.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that all flows will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the flows in model m:

```
MorphFlow.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged flows will be unblanked in
flag	Flag	Flag set on the flows that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the flows in model m flagged with f:

```
MorphFlow.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all flows will be unset in
flag	Flag	Flag to unset on the flows

Return type

No return value

Example

To unset the flag f on all the flows in model m:

```
MorphFlow.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the flow.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the flow is unsketched. If omitted redraw is true. If you want to unsketch several flows and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch flow flow:

```
flow.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flows.

Arguments

Name	Type	Description
Model	Model	Model that all flows will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the flows are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all flows in model m:

```
MorphFlow.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged flows in the model.

Arguments

Name	Type	Description
Model	Model	Model that all flows will be unsketched in
flag	Flag	Flag set on the flows that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the flows are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all flows flagged with flag in model m:

```
MorphFlow.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

MorphFlow object.

Example

To check if MorphFlow property flow.example is a parameter by using the MorphFlow.GetParameter() method:

```
if (flow.ViewParameters().GetParameter(flow.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for flow. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for flow flow:

```
flow.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this flow.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for flow flow:

```
var xrefs = flow.Xrefs();
```

---

toString()

Description

Creates a string containing the morph flow data in keyword format. Note that this contains the keyword header and the keyword cards. See also MorphFlow.Keyword() and MorphFlow.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for morph flow f in keyword format

```
var s = f.toString();
```

MorphPoint class

The MorphPoint class gives you access to morph points in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BlankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- First(Model*[Model]*)
- FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- FlagAll(Model*[Model]*, flag*[Flag]*)
- ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*)
- GetAll(Model*[Model]*)
- GetFlagged(Model*[Model]*, flag*[Flag]*)
- GetFromID(Model*[Model]*, number*[integer]*)
- Last(Model*[Model]*)
- LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- MoveFlagged(Model*[Model]*, flag*[Flag]*, dx*[real]*, dy*[real]*, dz*[real]*)
- NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*)
- Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*)
- RenumberAll(Model*[Model]*, start*[integer]*)
- RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*)
- Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*)
- SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- Total(Model*[Model]*, exists (optional)*[boolean]*)
- UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)
- UnflagAll(Model*[Model]*, flag*[Flag]*)
- UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*)
- UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*)

Member functions

- AssociateComment(Comment*[Comment]*)
- Blank()
- Blanked()
- ClearFlag(flag*[Flag]*)
- Copy(range (optional)*[boolean]*)
- DetachComment(Comment*[Comment]*)
- Error(message*[string]*, details (optional)*[string]*)
- Flagged(flag*[Flag]*)
- GetComments()
- GetParameter(prop*[string]*)
- Keyword()
- KeywordCards()
- Next()
- Previous()
- SetFlag(flag*[Flag]*)
- Sketch(redraw (optional)*[boolean]*)
- Unblank()
- Unsketch(redraw (optional)*[boolean]*)
- ViewParameters()
- Warning(message*[string]*, details (optional)*[string]*)
- Xrefs()
- toString()

MorphPoint properties

Name	Type	Description
exists	logical	true if morph point exists, false if referred to but not defined. (read only)
include	integer	The Include file number that the morph point is in.
label	integer	MorphPoint number.
model	integer	The Model number that the point is in.
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate

Detailed Description

The MorphPoint class allows you to create, modify and manipulate morph points. See the documentation below for more details.

Constructor

new MorphPoint(Model*[Model]*, label*[integer]*, x*[real]*, y*[real]*, z*[real]*)

Description

Create a new MorphPoint object.

Arguments

Name	Type	Description
Model	Model	Model that morph point will be created in
label	integer	MorphPoint number
x	real	X coordinate
y	real	Y coordinate
z	real	Z coordinate

Return type

MorphPoint object

Example

To create a new morph point in model m with label 100, at coordinates (20, 40, 10)

```
var n = new MorphPoint(m, 100, 20, 40, 10);
```

Details of functions

AssociateComment(Comment*[Comment]*)

Description

Associates a comment with a point.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be attached to the point

Return type

No return value

Example

To associate comment c to the point p:

```
p.AssociateComment(c);
```

Blank()

Description

Blanks the point

Arguments

No arguments

Return type

No return value

Example

To blank point p:

```
p.Blank();
```

BlankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all points will be blanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the points in model m:

```
MorphPoint.BlankAll(m);
```

BlankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Blanks all of the flagged points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged points will be blanked in
flag	Flag	Flag set on the points that you want to blank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To blank all of the points in model m flagged with f:

```
MorphPoint.BlankFlagged(m, f);
```

Blanked()

Description

Checks if the point is blanked or not.

Arguments

No arguments

Return type

true if blanked, false if not.

Example

To check if point p is blanked:

```
if (p.Blanked() ) do_something...
```

ClearFlag(flag[*Flag*])

Description

Clears a flag on the point.

Arguments

Name	Type	Description
flag	Flag	Flag to clear on the point

Return type

No return value

Example

To clear flag f for point p:

```
p.ClearFlag(f);
```

Copy(range (optional)*[boolean]*)

Description

Copies the point.

Arguments

Name	Type	Description
range (optional)	boolean	If you want to keep the copied item in the range specified for the current include. Default value is false. To set current include, use Include.MakeCurrentLayer().

Return type

MorphPoint object

Example

To copy point p into point z:

```
var z = p.Copy();
```

DetachComment(Comment*[Comment]*)

Description

Detaches a comment from a point.

Arguments

Name	Type	Description
Comment	Comment	Comment that will be detached from the point

Return type

No return value

Example

To detach comment c from the point p:

```
p.DetachComment(c);
```

Error(message*[string]*, details (optional)*[string]*)

Description

Adds an error for point. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The error message to give
details (optional)	string	An optional detailed error message

Return type

No return value

Example

To add an error message "My custom error" for point p:

```
p.Error("My custom error");
```

First(Model*[Model]*) [static]

Description

Returns the first point in the model.

Arguments

Name	Type	Description
Model	Model	Model to get first point in

Return type

MorphPoint object (or null if there are no points in the model).

Example

To get the first point in model m:

```
var p = MorphPoint.First(m);
```

FirstFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the first free point label in the model. Also see MorphPoint.LastFreeLabel(), MorphPoint.NextFreeLabel() and Model.FirstFreeItemLabel().

Arguments

Name	Type	Description
Model	Model	Model to get first free point label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *First free in layer* in editing panels). If omitted the whole model will be used (Equivalent to *First free* in editing panels).

Return type

MorphPoint label.

Example

To get the first free point label in model m:

```
var label = MorphPoint.FirstFreeLabel(m);
```

FlagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Flags all of the points in the model with a defined flag.

Arguments

Name	Type	Description
Model	Model	Model that all points will be flagged in
flag	Flag	Flag to set on the points

Return type

No return value

Example

To flag all of the points with flag f in model m:

```
MorphPoint.FlagAll(m, f);
```

Flagged(flag*[Flag]*)

Description

Checks if the point is flagged or not.

Arguments

Name	Type	Description
flag	Flag	Flag to test on the point

Return type

true if flagged, false if not.

Example

To check if point p has flag f set on it:

```
if (p.Flagged(f) ) do_something...
```

ForEach(Model*[Model]*, func*[function]*, extra (optional)*[any]*) [static]

Description

Calls a function for each point in the model.
Note that ForEach has been designed to make looping over points as fast as possible and so has some limitations. Firstly, a single temporary MorphPoint object is created and on each function call it is updated with the current point data. This means that you should not try to store the MorphPoint object for later use (e.g. in an array) as it is temporary.
Secondly, you cannot create new points inside a ForEach loop.

Arguments

Name	Type	Description
Model	Model	Model that all points are in
func	function	Function to call for each point
extra (optional)	any	An optional extra object/array/string etc that will appended to arguments when calling the function

Return type

No return value

Example

To call function test for all of the points in model m:

```
MorphPoint.ForEach(m, test);
function test(p)
{
// p is MorphPoint object
}
```

To call function test for all of the points in model m with optional object:

```
var data = { x:0, y:0 };
MorphPoint.ForEach(m, test, data);
function test(p, extra)
{
// p is MorphPoint object
// extra is data
}
```

GetAll(Model[*Model*]) [static]

Description

Returns an array of MorphPoint objects for all of the points in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get points from

Return type

Array of MorphPoint objects

Example

To make an array of MorphPoint objects for all of the points in model m

```
var p = MorphPoint.GetAll(m);
```

GetComments()

Description

Extracts the comments associated to a point.

Arguments

No arguments

Return type

$Array of Comment objects (or null if there are no comments associated to the node).

Example

To get the array of comments associated to the point p:

```
var comm_array = p.GetComments();
```

GetFlagged(Model*[Model]*, flag*[Flag]*) [static]

Description

Returns an array of MorphPoint objects for all of the flagged points in a model in Primer

Arguments

Name	Type	Description
Model	Model	Model to get points from
flag	Flag	Flag set on the points that you want to retrieve

Return type

Array of MorphPoint objects

Example

To make an array of MorphPoint objects for all of the points in model m flagged with f

```
var p = MorphPoint.GetFlagged(m, f);
```

GetFromID(Model*[Model]*, number*[integer]*) [static]

Description

Returns the MorphPoint object for a point ID.

Arguments

Name	Type	Description
Model	Model	Model to find the point in
number	integer	number of the point you want the MorphPoint object for

Return type

MorphPoint object (or null if point does not exist).

Example

To get the MorphPoint object for point 100 in model m

```
var p = MorphPoint.GetFromID(m, 100);
```

GetParameter(prop*[string]*)

Description

Checks if a MorphPoint property is a parameter or not. Note that object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. For this function to work the JavaScript interpreter must use the parameter name instead of the value. This can be done by setting the Options.property_parameter_names option to true before calling the function and then resetting it to false afterwards.. This behaviour can also temporarily be switched by using the MorphPoint.ViewParameters() method and 'method chaining' (see the examples below).

Arguments

Name	Type	Description
prop	string	point property to get parameter for

Return type

Parameter object if property is a parameter, null if not.

Example

To check if MorphPoint property p.example is a parameter:

```
Options.property_parameter_names = true;
if (p.GetParameter(p.example) ) do_something...
Options.property_parameter_names = false;
```

To check if MorphPoint property p.example is a parameter by using the GetParameter method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Keyword()

Description

Returns the keyword for this morph point (*MORPH_POINT). **Note that a carriage return is not added**. See also MorphPoint.KeywordCards()

Arguments

No arguments

Return type

string containing the keyword.

Example

To get the keyword for morph point p:

```
var key = p.Keyword();
```

KeywordCards()

Description

Returns the keyword cards for the morph point. **Note that a carriage return is not added**. See also MorphPoint.Keyword()

Arguments

No arguments

Return type

string containing the cards.

Example

To get the cards for morph point p:

```
var cards = p.KeywordCards();
```

Last(Model[*Model*]) [static]

Description

Returns the last point in the model.

Arguments

Name	Type	Description
Model	Model	Model to get last point in

Return type

MorphPoint object (or null if there are no points in the model).

Example

To get the last point in model m:

```
var p = MorphPoint.Last(m);
```

LastFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the last free point label in the model. Also see MorphPoint.FirstFreeLabel(), MorphPoint.NextFreeLabel() and see Model.LastFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get last free point label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest free in layer* in editing panels). If omitted the whole model will be used.

Return type

MorphPoint label.

Example

To get the last free point label in model m:

```
var label = MorphPoint.LastFreeLabel(m);
```

MoveFlagged(Model*[Model]*, flag*[Flag]*, dx*[real]*, dy*[real]*, dz*[real]*) [static]

Description

This function moves a selection of flagged morph points by a given vector and interpolates the movement of other morph points in the same way as this happens on the interactive morph panel. Note that the interpolation depends on the settings which can be switched on the interactive morph panel or by preferences. To apply the movement to the nodes in the box(es), you will need to call MorphBox.ApplyMorphing() at least for all relevant boxes or (if that is easier) for all morph boxes in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged morph points are in
flag	Flag	Flag set on the morph points explicitly selected to move
dx	real	X component of vector to be moved along
dy	real	Y component of vector to be moved along
dz	real	Z component of vector to be moved along

Return type

No return value

Example

To move all morph points in model m flagged with flag by 10 units in global Y direction while interpolating the other morph points as given by button settings or preferences:

```
MorphPoint.MoveFlagged(m, flag, 0.0, 10.0, 0.0);
```

Next()

Description

Returns the next point in the model.

Arguments

No arguments

Return type

MorphPoint object (or null if there are no more points in the model).

Example

To get the point in model m after point p:

```
var p = p.Next();
```

NextFreeLabel(Model*[Model]*, layer (optional)*[Include number]*) [static]

Description

Returns the next free (highest+1) point label in the model. Also see MorphPoint.FirstFreeLabel(), MorphPoint.LastFreeLabel() and Model.NextFreeItemLabel()

Arguments

Name	Type	Description
Model	Model	Model to get next free point label in
layer (optional)	Include number	Include file (0 for the main file) to search for labels in (Equivalent to *Highest+1 in layer* in editing panels). If omitted the whole model will be used (Equivalent to *Highest+1* in editing panels).

Return type

MorphPoint label.

Example

To get the next free point label in model m:

```
var label = MorphPoint.NextFreeLabel(m);
```

Pick(prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*, button text (optional)*[string]*) [static]

Description

Allows the user to pick a point.

Arguments

Name	Type	Description
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only points from that model can be picked. If the argument is a Flag then only points that are flagged with *limit* can be selected. If omitted, or null, any points from any model can be selected. from any model.
modal (optional)	boolean	If picking is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the pick will be modal.
button text (optional)	string	By default the window with the prompt will have a button labelled 'Cancel' which if pressed will cancel the pick and return null. If you want to change the text on the button use this argument. If omitted 'Cancel' will be used.

Return type

MorphPoint object (or null if not picked)

Example

To pick a point from model m giving the prompt 'Pick point from screen':

```
var p = MorphPoint.Pick('Pick point from screen', m);
```

Previous()

Description

Returns the previous point in the model.

Arguments

No arguments

Return type

MorphPoint object (or null if there are no more points in the model).

Example

To get the point in model m before point p:

```
var p = p.Previous();
```

RenumberAll(Model*[Model]*, start*[integer]*) [static]

Description

Renumbers all of the points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all points will be renumbered in
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the points in model m, from 1000000:

```
MorphPoint.RenumberAll(m, 1000000);
```

RenumberFlagged(Model*[Model]*, flag*[Flag]*, start*[integer]*) [static]

Description

Renumbers all of the flagged points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all the flagged points will be renumbered in
flag	Flag	Flag set on the points that you want to renumber
start	integer	Start point for renumbering

Return type

No return value

Example

To renumber all of the points in model m flagged with f, from 1000000:

```
MorphPoint.RenumberFlagged(m, f, 1000000);
```

Select(flag*[Flag]*, prompt*[string]*, limit (optional)*[Model or Flag]*, modal (optional)*[boolean]*) [static]

Description

Allows the user to select points using standard PRIMER object menus.

Arguments

Name	Type	Description
flag	Flag	Flag to use when selecting points
prompt	string	Text to display as a prompt to the user
limit (optional)	Model or Flag	If the argument is a Model then only points from that model can be selected. If the argument is a Flag then only points that are flagged with *limit* can be selected (*limit* should be different to *flag*). If omitted, or null, any points can be selected. from any model.
modal (optional)	boolean	If selection is modal (blocks the user from doing anything else in PRIMER until this window is dismissed). If omitted the selection will be modal.

Return type

Number of points selected or null if menu cancelled

Example

To select points from model m, flagging those selected with flag f, giving the prompt 'Select points':

```
MorphPoint.Select(f, 'Select points', m);
```

To select points, flagging those selected with flag f but limiting selection to points flagged with flag l, giving the prompt 'Select points':

```
MorphPoint.Select(f, 'Select points', l);
```

SetFlag(flag*[Flag]*)

Description

Sets a flag on the point.

Arguments

Name	Type	Description
flag	Flag	Flag to set on the point

Return type

No return value

Example

To set flag f for point p:

```
p.SetFlag(f);
```

Sketch(redraw (optional)*[boolean]*)

Description

Sketches the point. The point will be sketched until you either call MorphPoint.Unsketch(), MorphPoint.UnsketchAll(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the point is sketched. If omitted redraw is true. If you want to sketch several points and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch point p:

```
p.Sketch();
```

SketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Sketches all of the flagged points in the model. The points will be sketched until you either call MorphPoint.Unsketch(), MorphPoint.UnsketchFlagged(), Model.UnsketchAll(), or delete the model

Arguments

Name	Type	Description
Model	Model	Model that all the flagged points will be sketched in
flag	Flag	Flag set on the points that you want to sketch
redraw (optional)	boolean	If model should be redrawn or not after the points are sketched. If omitted redraw is true. If you want to sketch flagged points several times and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To sketch all points flagged with flag in model m:

```
MorphPoint.SketchFlagged(m, flag);
```

Total(Model*[Model]*, exists (optional)*[boolean]*) [static]

Description

Returns the total number of points in the model.

Arguments

Name	Type	Description
Model	Model	Model to get total for
exists (optional)	boolean	true if only existing points should be counted. If false or omitted referenced but undefined points will also be included in the total.

Return type

number of points

Example

To get the total number of points in model m:

```
var total = MorphPoint.Total(m);
```

Unblank()

Description

Unblanks the point

Arguments

No arguments

Return type

No return value

Example

To unblank point p:

```
p.Unblank();
```

UnblankAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all points will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the points in model m:

```
MorphPoint.UnblankAll(m);
```

UnblankFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unblanks all of the flagged points in the model.

Arguments

Name	Type	Description
Model	Model	Model that the flagged points will be unblanked in
flag	Flag	Flag set on the points that you want to unblank
redraw (optional)	boolean	If model should be redrawn or not. If omitted redraw is false. If you want to do several (un)blanks and only redraw after the last one then use false for all redraws apart from the last one. Alternatively you can redraw using View.Redraw().

Return type

No return value

Example

To unblank all of the points in model m flagged with f:

```
MorphPoint.UnblankFlagged(m, f);
```

UnflagAll(Model*[Model]*, flag*[Flag]*) [static]

Description

Unsets a defined flag on all of the points in the model.

Arguments

Name	Type	Description
Model	Model	Model that the defined flag for all points will be unset in

flag	Flag	Flag to unset on the points

Return type

No return value

Example

To unset the flag f on all the points in model m:

```
MorphPoint.UnflagAll(m, f);
```

Unsketch(redraw (optional)*[boolean]*)

Description

Unsketches the point.

Arguments

Name	Type	Description
redraw (optional)	boolean	If model should be redrawn or not after the point is unsketched. If omitted redraw is true. If you want to unsketch several points and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch point p:

```
p.Unsketch();
```

UnsketchAll(Model*[Model]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all points.

Arguments

Name	Type	Description
Model	Model	Model that all points will be unblanked in
redraw (optional)	boolean	If model should be redrawn or not after the points are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all points in model m:

```
MorphPoint.UnsketchAll(m);
```

UnsketchFlagged(Model*[Model]*, flag*[Flag]*, redraw (optional)*[boolean]*) [static]

Description

Unsketches all flagged points in the model.

Arguments

Name	Type	Description
Model	Model	Model that all points will be unsketched in
flag	Flag	Flag set on the points that you want to unsketch
redraw (optional)	boolean	If model should be redrawn or not after the points are unsketched. If omitted redraw is true. If you want to unsketch several things and only redraw after the last one then use false for redraw and call View.Redraw().

Return type

No return value

Example

To unsketch all points flagged with flag in model m:

```
MorphPoint.UnsketchAll(m, flag);
```

ViewParameters()

Description

Object properties that are parameters are normally returned as the integer or float parameter values as that is virtually always what the user would want. This function temporarily changes the behaviour so that if a property is a parameter the parameter name is returned instead. This can be used with 'method chaining' (see the example below) to make sure a property argument is correct.

Arguments

No arguments

Return type

MorphPoint object.

Example

To check if MorphPoint property p.example is a parameter by using the MorphPoint.GetParameter() method:

```
if (p.ViewParameters().GetParameter(p.example) ) do_something...
```

Warning(message*[string]*, details (optional)*[string]*)

Description

Adds a warning for point. For more details on checking see the Check class.

Arguments

Name	Type	Description
message	string	The warning message to give
details (optional)	string	An optional detailed warning message

Return type

No return value

Example

To add a warning message "My custom warning" for point p:

```
p.Warning("My custom warning");
```

Xrefs()

Description

Returns the cross references for this point.

Arguments

No arguments

Return type

Xrefs object.

Example

To get the cross references for point p:

```
var xrefs = p.Xrefs();
```

---

toString()

Description

Creates a string containing the morph point data in keyword format. Note that this contains the keyword header and the keyword cards. See also MorphPoint.Keyword() and MorphPoint.KeywordCards().

Arguments

No arguments

Return type

string

Example

To get data for morph point p in keyword format

```
var s = p.toString();
```

Options class

The Options class enables you to access several options in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Options constants

Constants for Promises

Name	Description
Options.RUN_PROMISE_CONSTRUCTOR	Allow/run promises when an API constructor is called
Options.RUN_PROMISE_METHOD	Allow/run promises when an API method is called
Options.RUN_PROMISE_PROPERTY	Allow/run promises when an API property getter/setter is done
Options.RUN_PROMISE_WINDOW_LOOP	Allow/run promises in a window event loop

Options class properties

Name	Type	Description
auto_confirm	logical	If true then PRIMER will automatically confirm (i.e. press the OK button) on (most) message boxes that are mapped. If false (default) then the message boxes will be shown and wait for the user to press a button. This option may be useful to help automate an operation where PRIMER would normally show a message box and wait for the user to press a button.
browse_ missing_ include_file	logical	If true (default) then PRIMER will popup a "BROWSE MISSING INCLUDE FILE" panel while reading the model. If false then it will throw an 'missing include file error' and continue reading the model.
dyna_version	string	The LS-DYNA version used to write keyword files. Can be "971R5", "971R4", "971R3", "970v6763" etc (see the version popup in Model->Write '>>> LS-Dyna output options' for a full list). See also Model.Write and Include.Write
edit_keep_ on_top	logical	If true edit panels created from the Edit() or Create() methods will be kept on top of other windows. If false (default) then they can be lowered.
exception_ messages	logical	If true (default) error messages will be printed to the dialogue box/stdout when an exception occurs in the API. If false they will not be printed. This option may be useful if you are using try/catch to manage exceptions and you do not want any error messages to be printed.
keyout_ binary	logical	If true then the output file will be written out in binary. If false (default) then an ascii file will be written.
keyout_ compress_ format	constant	This option can be used to specify the mode of compression. Can be Model.INDIVIDUAL_GZIP, Model.INDIVIDUAL_ZIP or Model.PACKAGED_ZIP

keyout_compress_level	integer	Compression level for .gz and .zip files. Must be in the range 1 to 9 with 1 being the least compression (fastest speed) to 9 being the greatest compression (slowest speed)
keyout_compress_switch	constant	Switch to set the compression during keyout. Can be Model.COMPRESS_KEEP (default), Model.COMPRESS_OFF or Model.COMPRESS_ON
keyout_i10	logical	If true then i10 format will be used to write the file. If false (default) then the normal LS-DYNA format will be used.
keyout_large	logical	If true then large format will be used to write the file. If false (default) then the normal LS-DYNA format will be used. Note that large format is only available from version R7.1 and above.
keyout_method	constant	The method used to write include files. Can be Include.MASTER_ONLY, Include.MERGE, Include.SELECT, Include.NOT_WRITTEN, Include.SUBDIR (default) or Include.SAME_DIR
keyout_parameter_values	logical	This option can be used to specify how parameters are written. If true then the underlying values of any parameters will be written when they are used in data fields rather than '&name'. If false (default) then '&name' will be written.
keyout_path_type	constant	The method used to write include paths. Can be Include.ABSOLUTE (default) or Include.RELATIVE
keyout_separator	constant	The directory separator used when writing include files. Can be Include.NATIVE (default), Include.UNIX or Include.WINDOWS
merge_set_collect	logical	If true then when merging models PRIMER will merge *SET_COLLECT cards which have the same label. If false (default) then they will be renumbered. This is also used with Model.ImportInclude. The default for this can be set using the primer*merge_set_collect preference.
model_tabs_active	logical	If true (default) then Primer will show model tabs in the object selection menu. If false then Primer will hide model tabs in object selection menu.
node_replace_asrg	logical	If true nodes in *AIRBAG_SHELL_REFERENCE_GEOMETRY can be replaced by node merge/replace. If false they will not be considered.
pick_window_position	constant or Window	Position that the pick window will be shown on the screen. It can be any combination (bitwise OR) of Window.LEFT, Window.CENTRE, Window.RIGHT, Window.TOP, Window.MIDDLE and Window.BOTTOM or a Window object. If a window obect is used the pick window will be shown in the middle of that window. The default is Window.RIGHT|Window.TOP.
property_parameter_names	logical	If true object properties which are parameters will be returned as parameter names. If false object properties which are parameters will be returned as parameter values.
reset_cwd	logical	If true then the current working directory will not be changed after selecting a file. If false (default) then the current working directory will be changed after selecting a file. This option only applies to Windows machines.
run_promises	constant	When any promise callbacks/handlers are allowed to run. Can be a bitwise OR of: Options.RUN_PROMISE_WINDOW_LOOP, Options.RUN_PROMISE_CONSTRUCTOR, Options.RUN_PROMISE_METHOD and Options.RUN_PROMISE_PROPERTY The default is for all to be allowed. Promise handlers can also be run manually by using Utils.CallPromiseHandlers()

Properties for connections

Name	Type	Description
connection_angle_tol	real	The angle tolerance used for spotwelds in the connections algorithm
connection_edge_dist	real	The edge distance used in the connections algorithm

connection_file	string	The connection file to read/write
connection_max_thickness	real	The maximum thickness used in the connections algorithm
connection_model	integer	The model number selected to make connections in
connection_part	integer	The part ID selected for connections
connection_write_flag (read only)	integer	Flag that will be set on selected connections when writing. This can be used in the user JavaScript to write connections to find which are selected.
solid_spotweld_diameter	real	The default diameter of solid spotwelds.
spotweld_element_type	integer	The default type of spotweld to make. can be: Conx.SPOTWELD_BEAM, Conx.SPOTWELD_SOLID1, Conx.SPOTWELD_SOLID4, Conx.SPOTWELD_SOLID8, Conx.SPOTWELD_SOLID12 or Conx.SPOTWELD_SOLID16

Properties for graphics

Name	Type	Description
airbag_colour	integer	Airbag symbol colour
background_colour	integer	Colour of the background
contacts_colour	integer	Contact surface colour
contour_text_size	integer	Contour bar text size
date_size	integer	Size of date (clock) display
edge_angle	real	Feature edge critical angle
edges_ign_pt	integer	Option for choosing how to draw free edges (can be set to TRUE or FALSE)
extra_nodes_colour	integer	Constrained extra nodes colour
feature_line	integer	Switch ON/OFF feature line (can be set to TRUE or FALSE)
for_mom_colour	integer	Nodal force/moment colour
graticule_text_size	integer	Graticule text size
label_colour	integer	Colour of the label
label_size	integer	Label size
node_colour	integer	Nodes colour
nrb_colour	integer	Nodal rigid body colour
overlay_colour	integer	Colour of the overlay
overlay_edges	integer	Option for setting the overlay edges value (can be set to 0,1 or 2)
rigid_bodies_colour	integer	Constrained rigid body colour
rot_vels_colour	integer	Rotational velocity colour
sketch_colour	integer	Colour of the sketch
spotweldbeam_colour_from_panels	integer	Spotweld beam/solid colour
spr_colour_from_node_sets	integer	Constrained SPR/SPR2/SPR3 colour
text_colour	integer	Colour of the text

timehist_blks_colour	integer	Time history block colour
tracer_partl_colour	integer	Tracer particle colour
trans_vels_colour	integer	Translational velocity colour
x_sections_colour	integer	Cross-section colour

Properties for mass properties calculation

Name	Type	Description
mass_properties_centre_x	real	X-coordinate of user defined centre.
mass_properties_centre_y	real	Y-coordinate of user defined centre.
mass_properties_centre_z	real	Z-coordinate of user defined centre.
mass_properties_coordinate_system_type	integer	Coordinate system selection: Model.GLOBAL_AXES, Model.LOCAL_AXES, Model.PRINCIPAL_AXES.
mass_properties_include_attached_mass_deformable_elems	logical	Option to include lumped mass attached to the nodes of deformable elements. Default is FALSE.
mass_properties_include_attached_mass_rigid_elems	logical	Option to include lumped mass attached to the nodes of rigid elements. Default is FALSE.
mass_properties_include_timestep_mass	logical	Option to switch on/off inclusion of timestep addded mass. Default is FALSE.
mass_properties_inertia_center	integer	Option to set the centre used in inertia properties calculation. By default Centre at CofG is used. Available options are: Model.CENTRE_AT_COFG, Model.USER_DEFINED_CENTRE.
mass_properties_local_axes	integer	CSYS ID when using local axes.
mass_properties_rigid_part_extra_nodes	logical	Option to switch on/off mass of *CONSTRAINED_EXTRA_NODES associated with a rigid part. Default is FALSE.
mass_properties_rigid_part_slave_parts	logical	Option to switch on/off mass of *CONSTRAINED_RIGID_BODIES associated with a rigid part. Default is FALSE.

Properties for nastran

Name	Type	Description
convert_rbe2_cnrb	logical	Convert all RBE2s to *CONSTRAINED_NODAL_RIGID_BODY
merge_rbe_nodes	logical	Merge duplicate RBE dependent nodes
retain_mid_nodes	logical	Retain mid-side nodes for higher order elements

Properties for ssh

Name	Type	Description
ssh_buffer_size	integer	The size of the buffer used (in kiloBytes) when transferring data to/from the remote machine in the Ssh class. Depending on your network and the size of the files you are transferring, changing this value may make file transfers quicker. The default value is 64(kB) but any value in the range 1(kB) to 1024(kB) is allowed.

Properties for widgets

Name	Type	Description

max_widgets	integer	The maximum number of Widgets that can be made for one Window. The default value is 1000
max_ window_ lines	integer	The maximum number of lines that can be made for a Window.Error(), Window.Information(), Window.Message(), Window.Question() or Window.Warning() window. The default value is 25

Detailed Description

The Options class is used to get/set options that PRIMER uses for certain functions. The options are available as **class** properties. See the documentation for more details. An example: Options.mass_properties_include_attached_mass_ deformable_elems=true

PopupWindow class

The PopupWindow class allows you to create popup windows for a graphical user interface. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- Hide()

PopupWindow properties

Name	Type	Description
persistent	boolean	If the popup window will remain mapped when a button is pressed in it. By default (false) when a button is pressed in the popup window the popup will be unmapped. If set to true then the popup will remain mapped until the user clicks out of the window or hides it by calling Hide()

Detailed Description

The PopupWindow class allows you to make popup windows (that you can place Widgets in) and link them to Widgets. The popup window is then displayed by right clicking on the Widget the popup is linked to. The following very simple example shows how to create a popup window and link it to a label Widget.

```
// Create popup window
var pw = new PopupWindow();
// Create some widgets in the popup window
var pl = new Widget(pw, Widget.LABEL,   1, 30,  1,  7, "Label");
var pb = new Widget(pw, Widget.BUTTON,  1, 30,  7, 13, "Button");
var pt = new Widget(pw, Widget.TEXTBOX, 1, 30, 20, 26, "Textbox");
// Create window with title "Popup example" from 0.8-1.0 in x and 0.5-0.6 in y
var w = new Window("Popup example", 0.8, 1.0, 0.5, 0.6);
// Create label widget
var l = new Widget(w, Widget.LABEL,   1, 50, 1,  7, "Right click for popup...");
// link popup window to widget
l.popupWindow = pw;
// Assign the onPopup callback method to the function 'do_popup'
// This is only required if you want to make any changes before the popup
appears
l.onPopup  = do_popup;
// Show the widget and start event loop
w.Show();
////////////////////////////////////////////////////////////////////////////
function do_popup()
{
    Message("Showing popup");
}
```

See the documentation below and the Widget class for more details.

Constructor

new PopupWindow()

Description

Create a new PopupWindow object.

Arguments

No arguments

Return type

PopupWindow object

Example

To create a PopupWindow containing the buttons "Create" and "Edit" and link it to button b:

```
var pw = new PopupWindow();
var c = new Widget(pw, Widget.BUTTON, 1, 30, 1, 7,  "Create");
var e = new Widget(pw, Widget.BUTTON, 1, 30, 7, 13, "Edit");
b.popupWindow = pw;
```

Details of functions

Hide()

Description

Hides (unmaps) the popup window.

Arguments

No arguments

Return type

No return value

Example

To hide popup window w:

```
w.Hide();
```

Ssh class

The Ssh class allows you to connect to a remote computer using ssh, scp and sftp commands. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- AuthenticateWithPassword(password*[string]*)
- AuthenticateWithPublicKey(passphrase (optional)*[string]*)
- Execute(data*[object]*)
- Get(remote*[string]*, local*[string]*)
- Put(remote*[string]*, local*[string]*)
- SftpGet(remote*[string]*, local*[string]*)
- SftpList(remote*[string]*)
- SftpMkdir(remote*[string]*, mode*[constant]*)
- SftpPut(remote*[string]*, local*[string]*)
- SftpRmdir(remote*[string]*)

Ssh constants

Constants for file bits

Name	Description
Ssh.SETGROUP_BIT	Set group bit
Ssh.SETUID_BIT	Set uid bit
Ssh.STICKY_BIT	sticky bit

Constants for file types

Name	Description
Ssh.DIRECTORY	Directory
Ssh.FILE	Regular file
Ssh.SOCKET	Socket
Ssh.SYMBOLIC_LINK	Symbolic link

Constants for permissions

Name	Description
Ssh.GROUP_EXECUTE	Group has execute permission
Ssh.GROUP_READ	Group has read permission
Ssh.GROUP_WRITE	Group has write permission
Ssh.OTHER_EXECUTE	Others have execute permission
Ssh.OTHER_READ	Others have read permission

Ssh.OTHER_WRITE	Others have write permission
Ssh.OWNER_EXECUTE	Owner has execute permission
Ssh.OWNER_READ	Owner has read permission
Ssh.OWNER_WRITE	Owner has write permission

Detailed Description

The Ssh class gives you simple functions to do secure connections to a remote computer using ssh. The Oasys Ltd LS-DYNA environment software is built with the OpenSSH library to support the ssh, scp and sftp protocols. The basic workflow is to create a connection using the Ssh constructor, authenticate the connection either by using a password and AuthenticateWithPassword or with a public key and AuthenticateWithPublicKey then the method Execute can be used to execute commands on the remote machine, the methods Get and Put can be used to copy files to and from the remote machine using scp, and the commands SftpGet, SftpList, SftpMkdir, SftpPut and SftpRmdir can be used to perform secure file transfer commands.
ssh uses a public and private key pair to do communication. The software uses RSA for the private and public keys and stores them in the files id_rsa and id_rsa.pub in the .oasys_ssh directory of your home directory (C:\Users\your.name\.oasys_ssh on Windows by default). A key length of 2048 bits is recommended. You keep your private key secure in your .oasys_ssh directory but the public key can be copied to the authorized_keys file on remote machines so that authentication can be done etc. The software also maintains fingerprints for the machines you connect to to ensure that you are connecting to the machine that you think you are. The first time you connect to a machine you are asked to confirm the remote machine is correct and the software stores the fingerprint for it in the known_hosts file in your .oasys_ssh directory. For second and subsequent connections the software checks the fingerprint of the remote machine against the one it has stored and will only connect if it matches.
When creating a new ssh connection to a remote machine and transferring files a small 'buffer' is required to transfer the data. The size of this buffer can be controlled using the Options.ssh_buffer_size property **before** the Ssh object is created.

Constructor

new Ssh(hostname*[string]*, username*[string]*)

Description

Create a new Ssh object for secure communication to a remote computer.

Arguments

Name	Type	Description
hostname	string	The hostname of the machine that you want to connect to
username	string	The username on the machine that you want to connect to

Return type

Ssh object

Example

To create a connection to machine "example" as user "username"

```
var s = new Ssh("example", "username");
```

Details of functions

AuthenticateWithPassword(password*[string]*)

Description

Authenticate the connection using password.

Arguments

Name	Type	Description
password	string	The password for the username on the remote machine

Return type

no return value

Example

To prompt the user for a password and authenticate using it in SSH connection s:

```
var password = Window.GetPassword("Enter Password to connect", "Password");
s.AuthenticateWithPassword(password);
```

AuthenticateWithPublicKey(passphrase (optional)*[string]*)

Description

Authenticate the connection using your public key. Your public key from the file .oasys_ssh/id_rsa.pub must be in the file .oasys_ssh/authorized_keys on the remote machine.

Arguments

Name	Type	Description
passphrase (optional)	string	The passphrase for authentication on the remote machine if required

Return type

no return value

Example

Authenticate using your public key in SSH connection s:

```
s.AuthenticateWithPublicKey();
```

Execute(data*[object]*)

Description

Execute a command in the ssh session and get the standard output and error streams.

Arguments

Name	Type	Description
data	object	Execute data Object has the following properties:

Name	Type	Description
arguments (optional)	Array of strings	The arguments to pass to the command
command	string	The command you want to run

Return type

Object with the following properties:

Name	Type	Description
status	integer	The exit code from the command
stderr	string	The standard error output from the command

stdout	string	The standard output from the command

Example

To run command "example.bat" with arguments "foo" and "bar" in SSH connection s:

```
var output = s.Execute( { command: 'example.bat', arguments: [ 'foo', 'bar' ]}
);
var text   = output.stdout;
var errors = output.stderr;
var ecode  = output.status;
```

Get(remote*[string]*, local*[string]*)

Description

Gets a file from the ssh connection using scp.

Arguments

Name	Type	Description
remote	string	The path of the remote file to get
local	string	The path of the local file to write

Return type

no return value

Example

To get the remote file "/path/to/file.txt", creating local file "C:\path\to\file.txt" in SSH connection s:

```
s.Get("/path/to/file.txt", "C:\\path\\to\\file.txt");
```

Put(remote*[string]*, local*[string]*)

Description

Puts a file on the remote ssh connection using scp.

Arguments

Name	Type	Description
remote	string	The path of the remote file to put
local	string	The path of the local file to read

Return type

no return value

Example

To put the local file "C:\path\to\file.txt" to remote file "/path/to/file.txt" in SSH connection s:

```
s.Put("/path/to/file.txt", "C:\\path\\to\\file.txt");
```

SftpGet(remote*[string]*, local*[string]*)

Description

Gets a file from the ssh connection using sftp.

Arguments

Name	Type	Description
remote	string	The path of the remote file to get
local	string	The path of the local file to write

Return type

no return value

Example

To get the remote file "file.txt", creating local file "C:\path\to\file.txt" in SSH connection s using sftp:

```
s.SftpGet("file.txt", "C:\\path\\to\\file.txt");
```

SftpList(remote*[string]*)

Description

Gets a listing from the ssh connection using sftp.

Arguments

Name	Type	Description
remote	string	The remote path to get the listing from

Return type

Object with the following properties:

Name	Type	Description
atime	integer	Access time for the file (seconds since epoch)
gid	integer	The group ID
info	constant	Bitwise information for the file/directory. See the permissions, file types and file bits constants
mtime	integer	Modification time for the file (seconds since epoch)
name	string	The name of the file/directory
size	integer	The size of the file
uid	integer	The user ID

Example

To get listing from the the remote path "temp" in SSH connection s using sftp:

```
var listing = s.SftpList("temp");
for (l=0; l<listing.length; l++)
{
    Message(listing[l].name + ":" + listing[l].size;
}
```

SftpMkdir(remote*[string]*, mode*[constant]*)

Description

Creates a directory in the remote ssh connection using sftp.

Arguments

Name	Type	Description
remote	string	The remote directory to create
mode	constant	The mode/permissions for the directory. See the permissions constants for details. Note that the user's file-creation mask (umask) value will also be taken into account when creating the directory.

Return type

true if successful, false if not

Example

To create the remote path "temp" with, create, write and execute permissions for only the owner, in SSH connection s using sftp:

```
var success = s.SftpMkdir("temp", Ssh.OWNER_READ|Ssh.OWNER_WRITE|Ssh.OWNER_
EXECUTE);
```

SftpPut(remote*[string]*, local*[string]*)

Description

Puts a file on the remote ssh connection using sftp.

Arguments

Name	Type	Description
remote	string	The path of the remote file to put
local	string	The path of the local file to read

Return type

no return value

Example

To put the local file "C:\path\to\file.txt" to remote file "file.txt" in SSH connection s:

```
s.SftpPut("file.txt", "C:\\path\\to\\file.txt");
```

SftpRmdir(remote*[string]*)

Description

Deletes a directory in the remote ssh connection using sftp. If this fails it is probably because the directory is not empty.

Arguments

Name	Type	Description
remote	string	The remote directory to delete

Return type

true if successful, false if not

Example

To delete the remote path "temp" in SSH connection s using sftp:

```
var success = s.SftpRmdir("temp");
```

Utils class

The Utils class contains various useful utility functions. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Ascii85Decode(encoded*[string]*)
- Ascii85Encode(data*[ArrayBuffer]*, length (optional)*[integer]*)
- Build()
- CallPromiseHandlers()
- CheckinLicense(feature*[string]*)
- CheckoutLicense(feature*[string]*)
- GarbageCollect()
- HTMLBrowser()
- HiResTimer()
- PdfReader()
- TimerResolution()
- Version()

Detailed Description

The Utils class is used to provide various useful functions.

Details of functions

Ascii85Decode(encoded*[string]*) [static]

Description

Decodes an ASCII85 encoded string. See Utils.Ascii85Encode() for details on the method.

Arguments

Name	Type	Description
encoded	string	An ASCII85 encoded string

Return type

ArrayBuffer object

Example

To decode an ASCII85 encoded string:

```
var decoded = Utils.Ascii85Decode(encoded);
```

Ascii85Encode(data*[ArrayBuffer]*, length (optional)*[integer]*) [static]

Description

Encodes an ASCII85 encoded string. This enables binary data to be represented by ASCII characters using five ASCII characters to represent four bytes of binary data (making the encoded size 1/4 larger than the original). By doing this binary data can be stored in JavaScript strings. Note that the method used by PRIMER to encode and decode strings differs from the standard ASCII85 encoding as that uses the ASCII characters ", ’ and \ which cannot be used in JavaScript strings as they have special meanings. The method in PRIMER uses
0-84 are !-u (ASCII codes 33-117) (i.e. 33 is added to it) with the following exceptions
v is used instead of " (ASCII code 118 instead of 34)
w is used instead of ’ (ASCII code 119 instead of 39)
x is used instead of \ (ASCII code 120 instead of 92)
If all five digits are 0 they are represented by a single character z instead of !!!!!

Arguments

Name	Type	Description
data	ArrayBuffer	ArrayBuffer containing the data
length (optional)	integer	Length of data in array buffer to encode. If omitted the whole array buffer will be encoded

Return type

string

Example

To encode ArrayBuffer data:

```
var encoded = Utils.Ascii85Encode(data);
```

Build() [static]

Description

Returns the build number

Arguments

No arguments

Return type

integer

Example

To get the current build number

```
var build = Utils.Build();
```

CallPromiseHandlers() [static]

Description

Manually call any promise handlers/callbacks in the job queue

Arguments

No arguments

Return type

no return value

Example

To run any queued promise handlers/callbacks:

```
Utils.CallPromiseHandlers();
```

CheckinLicense(feature*[string]*) [static]

Description

Checks a license for a feature back in

Arguments

Name	Type	Description
feature	string	feature to check license back in for

Return type

no return value

Example

To check in a license for "EXAMPLE":

```
Utils.CheckinLicense("EXAMPLE");
```

CheckoutLicense(feature*[string]*) [static]

Description

Checks out a license for a feature

Arguments

Name	Type	Description
feature	string	feature to check license for

Return type

true if license available, false if not

Example

To checkout a license for "EXAMPLE":

```
var got = Utils.CheckoutLicense("EXAMPLE");
if (got == false) Exit();
```

GarbageCollect() [static]

Description

Forces garbage collection to be done. This should not normally need to be called but in exceptional circumstances it can be called to ensure that garbage collection is done to return memory.

Arguments

No arguments

Return type

no return value

Example

To force garbage collection to be done:

```
Utils.GarbageCollect();
```

HTMLBrowser() [static]

Description

Returns the path to the default HTML browser

Arguments

No arguments

Return type

string of the path

Example

To get path to the default HTML browser

```
var path = Utils.HTMLBrowser();
```

HiResTimer() [static]

Description

A high resolution timer that can be used to time how long things take. The first time this is called the timer will start and return 0. Subsequent calls will return the time in nanoseconds since the first call. Note that the timer will almost certainly not have 1 nanosecond precision but, depending on the platform, should should have a resolution of at least 1 microsecond. The resolution can be found by using Utils.TimerResolution()

Arguments

No arguments

Return type

number

Example

To time how long something takes to nanosecond precision:

```
var start = Utils.HiResTimer();
do something that takes some time...
var end = Utils.HiResTimer();
Message("it took " + (end-start) + "nanoseconds");
```

PdfReader() [static]

Description

Returns the path to the executable of the default pdf reader

Arguments

No arguments

Return type

string of the path

Example

To get path to the default pdf reader

```
var path = Utils.PdfReader();
```

TimerResolution() [static]

Description

Returns the resolution (precision) of the Utils.HiResTimer() timer in nanoseconds

Arguments

No arguments

Return type

number

Example

To find the resolution of the timer in nanoseconds:

```
var resolution = Utils.TimerResolution();
```

Version() [static]

Description

Returns the version number

Arguments

No arguments

Return type

real

Example

To get the current version number

```
var version = Utils.Version();
```

View class

The View class allows you to control the view and plotting modes in PRIMER. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- Ac()
- Ct()
- Hi()
- Li()
- Redraw()
- SetContourType(View type*[constant]*, View subtype*[constant]*)
- Sh()
- Show(View type*[constant]*)
- Si()
- Vec()

View constants

Constants for SetContourType - argument 1 (Type)

Name	Description
View.ELEMPROPS	Type Element Properties
View.ELEMQUAL	Type Element Quality
View.INITVELS	Type Initial Velocities
View.LOADSHELLDIRECTION	Type Load Shell Direction
View.MASSSCALE	Type Mass Scale
View.MATLPROPS	Type Material Properties
View.PARTMASS	Type Part Mass
View.SHELLNORMALS	Type Shell Normals
View.SHELLTHICKNESS	Type Shell Thickness
View.TIMESTEP	Type TimeStep

Constants for SetContourType - argument 2 (Subtype)

Name	Description
View.ABSOLUTE	Subtype Absolute (of type Shell Thickness)
View.ADDEDMASS	Subtype Added Mass (of type Mass Scale)
View.ADDEDMASSPART	Subtype Added Mass #Part (of type Mass Scale)
View.AREA	Subtype Area (2d only) (of type Element Property)
View.ASPECTRATIO	Subtype Aspect Ratio (of type Element Quality)

View.CONTOUR	Subtype Contour (of type Shell Normals)
View.DENSITY	Subtype Density (of type Material Property)
View.EMPFINALMASS	Subtype (*)Final Mass (of type (*)EMP Parts Only)
View.EMPNSMASS	Subtype (*)NS Mass (of type (*)EMP Parts Only)
View.EMPSTRUCTMASS	Subtype (*)Struct Mass (of type (*)EMP Parts Only)
View.FAILEDCRITERIA	Subtype Failed Criteria (of type Element Quality)
View.FINALMASS	Subtype Final Mass (of type (*)EMP Parts Only)
View.FORM	Subtype Form (native) (of type Element Property)
View.FORMULATION	Subtype Formulation (of type Element Property)
View.INITVELRES	Subtype Init Vel-Res (of type Init Vel Component)
View.INITVELX	Subtype Init Vel-X (of type Init Vel Component)
View.INITVELY	Subtype Init Vel-Y (of type Init Vel Component)
View.INITVELZ	Subtype Init Vel-Z (of type Init Vel Component)
View.INTPOINTS	Subtype Integration Points (of type Element Property)
View.JACOBIAN	Subtype Jacobian (of type Element Quality)
View.MATERIALNUMBER	Subtype Material Number (of type Material Property)
View.MAXINTANGLE	Subtype Max Internal Angle (of type Element Quality)
View.MININTANGLE	Subtype Min Internal Angle (of type Element Quality)
View.MINLENGTH	Subtype Min Length (of type Element Quality)
View.PERCENTADDEDMASS	Subtype % Added Mass (of type Mass Scale)
View.PERCENTADDEDMASSPART	Subtype % Added Mass #Part (of type Mass Scale)
View.PLASTICSTRAIN	Subtype Plastic Strain (of type Element Property)
View.POISSONRATIO	Subtype Poisson’s Ratio (of type Material Property)
View.QUALIMPERF	Subtype Tet Collapse (of type Element Quality)
View.REMAINING	Subtype % remaining (of type Shell Thickness)
View.SKEW	Subtype Skew (native) (of type Element Quality)
View.STRUCTMASS	Subtype Struct Mass (of type (*)EMP Parts Only)
View.TAPER	Subtype Taper (of type Element Quality)
View.TETCOLLAPSE	Subtype Formulation (of type Element Quality)
View.THINNING	Subtype % thinning (of type Shell Thickness)
View.VECTOR	Subtype Vector (of type Shell Normals)
View.VOLUME	Subtype Volume (of type Element Property)
View.WARPAGE	Subtype Warpage (of type Element Quality)
View.YIELDSTRESS	Subtype Yield Stress (of type Material Property)
View.YOUNGMODULUS	Subtype Young’s Modulus (of type Material Property)

Constants for Show

Name	**Description**
View.ISO	Isometric projection

View.XY	XY axis projection
View.XZ	XZ axis projection
View.YZ	YZ axis projection

Detailed Description

The View class gives you access to the different plotting modes and views. See the documentation below for more details.

Details of functions

Ac() [static]

Description

Autoscales the view

Arguments

No arguments

Return type

No return value

Example

To autoscale

```
View.Ac();
```

Ct() [static]

Description

Does a contour plot

Arguments

No arguments

Return type

No return value

Example

To do a contour plot

```
View.Ct();
```

Hi() [static]

Description

Does a Hidden line plot

Arguments

No arguments

Return type

No return value

Example

To do a hidden line plot

```
View.Hi();
```

Li() [static]

Description

Does a line (wireframe) plot

Arguments

No arguments

Return type

No return value

Example

To do a line plot

```
View.Li();
```

Redraw() [static]

Description

Redraws the plot using the current plot mode.

Arguments

No arguments

Return type

No return value

Example

To redraw

```
View.Redraw();
```

SetContourType(View type*[constant]*, View subtype*[constant]*) [static]

Description

Sets a contour type (and subtype)

Arguments

Name	Type	Description

View type	constant	The type of contour to plot. Can be: View.TIMESTEP View.SHELLTHICKNESS View.SHELLNORMALS View.LOADSHELLDIRECTION View.ELEMPROPS View.ELEMQUAL View.MASSSCALE View.MATLPROPS View.INITVELS View.PARTMASS
View subtype	constant	The subtype of contour to plot. Note: This second argument is NOT required for types TIMESTEP and LOADSHELLDIRECTION. Subtypes for Type TIMESTEP: No subtypes Subtypes for Type SHELLTHICKNESS: View.ABSOLUTE View.THINNING View.REMAINING Subtypes for SHELLNORMALS: View.CONTOUR View.VECTOR Subtypes for Type LOADSHELLDIRECTION: No subtypes Subtypes for Type ELEMPROPS: View.FORMULATION View.INTPOINTS View.PLASTICSTRAIN View.FORM View.AREA View.VOLUME Subtypes for Type ELEMQUAL: View.MINLENGTH View.ASPECTRATIO View.WARPAGE View.SKEW View.MININTANGLE View.MAXINTANGLE View.JACOBIAN View.TAPER View.TETCOLLAPSE View.QUALIMPERF View.FAILEDCRITERIA Subtypes for Type MASSSCALE: View.PERCENTADDEDMASS View.ADDEDMASS View.PERCENTADDEDMASSPART View.ADDEDMASSPART Subtypes for Type MATLPROPS: View.DENSITY View.YIELDSTRESS View.POISSONRATIO View.YOUNGMODULUS View.MATERIALNUMBER Subtypes for Type INITVELS: View.INITVELX View.INITVELY View.INITVELZ View.INITVELRES Subtypes for Type PARTMASS: View.STRUCTMASS View.EMPSTRUCTMASS View.EMPNSMASS View.FINALMASS View.EMPFINALMASS

Return type

No return value

Example

To set a contour plot of Load Shell Direction (no subtype):

```
View.SetContourType(View.LOADSHELLDIRECTION);
```

To set a contour plot of Element Formulation (type: Elem Props, subtype: Formulation):

```
View.SetContourType(View.ELEMPROPS, View.FORMULATION);
```

Sh() [static]

Description

Does a shaded plot

Arguments

No arguments

Return type

No return value

Example

To do a shaded plot

```
View.Sh();
```

Show(View type*[constant]*) [static]

Description

Redraws using one of the standard views

Arguments

Name	Type	Description
View type	constant	The view to show. Can be +/-View.XY, +/-View.YZ, +/-View.XZ or +/-View.ISO

Return type

No return value

Example

To do an isometric view from the negative direction:

```
View.Show(-View.ISO);
```

Si() [static]

Description

Does a shaded image contour plot

Arguments

No arguments

Return type

No return value

Example

To do a shaded image contour plot

```
View.Si();
```

Vec() [static]

Description

Does a vector plot

Arguments

No arguments

Return type

No return value

Example

To do a vector plot

```
View.Vec();
```

Widget class

The Widget class allows you to create components for a graphical user interface. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- CtrlPressed()
- PixelsPerUnit()
- ShiftPressed()
- StringLength(text*[string]*, monospace (optional)*[boolean]*, fontSize (optional)*[integer]*)

Member functions

- AddWidgetItem(item*[WidgetItem]*, position (optional)*[integer]*)
- Circle(colour*[constant]*, fill*[boolean]*, xc*[integer]*, yc*[integer]*, radius*[integer]*)
- Clear()
- ClearSelection()
- Cross(colour (optional)*[constant]*)
- Delete()
- DirectoryIcon(line_colour*[constant]*, fill_colour*[constant]*)
- DumpImageString(filename*[string]*, format (optional)*[constant]*)
- Hide()
- ItemAt(index*[integer]*)
- Line(colour*[constant]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)
- Polygon(colour*[constant]*, fill*[boolean]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)
- ReadImageFile(filename*[string]*, justify (optional)*[constant]*, transparent (optional)*[colour value (integer)]*, tolerance (optional)*[integer]*)
- ReadImageString(string*[string]*, justify (optional)*[constant]*, transparent (optional)*[colour value (integer)]*, tolerance (optional)*[integer]*)
- Rectangle(colour*[constant]*, fill*[boolean]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)
- RemoveAllWidgetItems()
- RemoveWidgetItem(item*[WidgetItem]*)
- Show()
- Static()
- Tick(colour (optional)*[constant]*)
- TotalItems()
- WidgetItems()

Widget constants

Name	Description
Widget.BUTTON	Button widget
Widget.CHECKBOX	Checkbox widget
Widget.COMBOBOX	Combobox widget
Widget.LABEL	Label widget
Widget.LISTBOX	Listbox widget
Widget.SLIDER	Slider widget
Widget.TEXTBOX	Text input widget

Constants for Colour

Name	Description
Widget.BLACK	Colour black
Widget.BLUE	Colour blue
Widget.COLOUR_ CONTRAST	A contrasting colour in the 3 user interface themes (Green, Purple, and Blue in the Dark, Light, and Classic themes respectively). Blue in the legacy theme.
Widget.COLOUR_ CONTRAST_2	Another contrasting colour in the 3 user interface themes (Yellow, Red, and Red in the Dark, Light, and Classic themes respectively). Red in the legacy theme.
Widget.COLOUR_ INVERSE	Inverse colour in the 3 user interface themes (Black or white depending on theme). Black in the legacy theme.
Widget.COLOUR_LABEL	Label text colour in the 3 user interface themes (Black or white depending on theme). Black in the legacy theme.
Widget.COLOUR_ NEUTRAL	Neutral colour in the 3 user interface themes (Different shade of grey in every theme). Light grey in the legacy theme.
Widget.COLOUR_SAFE	Safe colour in the 3 user interface themes (Different shade of green in every theme). Dark green in the legacy theme.
Widget.COLOUR_TITLE	Title colour in the 3 user interface themes (Different shade of grey in every theme). Dark blue in the legacy theme.
Widget.COLOUR_ WARNING	Warning colour in the 3 user interface themes (Different shade of red in every theme). Dark red in the legacy theme.
Widget.CYAN	Colour cyan
Widget.DARKBLUE	Colour dark blue
Widget.DARKGREEN	Colour dark green
Widget.DARKGREY	Colour dark grey
Widget.DARKGREY_ NEUTRAL	Only valid in the function ’Line’. Used to keep the 3D effect in the legacy theme and not in the other themes. Neutral colour in the 3 user interface themes (Different shade of grey in every theme). Dark grey in the legacy theme
Widget.DARKRED	Colour dark red
Widget.DEFAULT	Default colour for widgets
Widget.GREEN	Colour green
Widget.GREY	Colour grey
Widget.LIGHTGREY	Colour light grey
Widget.LIGHTGREY_ NEUTRAL	Only valid in the function ’Line’. Used to keep the 3D effect in the legacy theme and not in the other themes. Neutral colour in the 3 user interface themes (Different shade of grey in every theme). Light grey in the legacy theme
Widget.MAGENTA	Colour magenta
Widget.ORANGE	Colour orange
Widget.RED	Colour red
Widget.WHITE	Colour white
Widget.YELLOW	Colour yellow

Constants for Image RGB format

Name	Description

Widget.RGB24	24 bits for RGB data in widget images
Widget.RGB8	8 bits for RGB data in widget images

Constants for Justification

Name	Description
Widget.BOTTOM	Bottom justification
Widget.CENTRE	Centre (horizontal) justification
Widget.LEFT	Left justification
Widget.MIDDLE	Middle (vertical) justification
Widget.RIGHT	Right justification
Widget.SCALE	Image will be scaled to fit widget
Widget.TOP	Top justification

Constants for Orientation

Name	Description
Widget.HORIZONTAL	Horizontal orientation (for sliders)
Widget.VERTICAL	Vertical orientation (for sliders)

Constants for Selection

Name	Description
Widget.SELECT_ENHANCED	Multiple WidgetItems in a ListBox Widget can be selected. When the user selects a WidgetItem the selection is cleared and the new WidgetItem selected. However, if the user presses the Ctrl key when clicking on a WidgetItem, the clicked WidgetItem gets toggled and all other WidgetItems are left untouched. If the user presses the Shift key while clicking on a WidgetItem, all WidgetItems between the last selected WidgetItem and the clicked WidgetItem are selected or unselected, depending on the state of the clicked WidgetItem.
Widget.SELECT_MULTIPLE	Multiple WidgetItems in a ListBox Widget can be selected. When the user selects a WidgetItem, the selection status of that WidgetItem is toggled and the other WidgetItems are left alone.
Widget.SELECT_NONE	No WidgetItem in a ListBox Widget can be selected
Widget.SELECT_SINGLE	A single WidgetItem in a ListBox Widget can be selected. When the user selects a WidgetItem, any already-selected WidgetItem becomes unselected, and the user cannot unselect the selected WidgetItem by clicking on it.

Constants for User interface categories

Name	Description
Widget.CATEGORY_APPLY	Apply buttons
Widget.CATEGORY_BUTTON_BOX	A button box panel that contains other widgets
Widget.CATEGORY_CANCEL	Buttons which cancel the current operation
Widget.CATEGORY_DATA_ENTRY_HEADER	Header for data entry cells, e.g. PRIMER create panels
Widget.CATEGORY_DISMISS	Buttons to close or dismiss panels
Widget.CATEGORY_ENTITY	Entity types in T/HIS

Widget.CATEGORY_GENERIC	A generic button that isn't a special category
Widget.CATEGORY_GENERIC_2	An alternative to the generic category that has a complementary colour
Widget.CATEGORY_HELP	Help buttons
Widget.CATEGORY_KEYWORD	A PRIMER keyword button
Widget.CATEGORY_LABEL	A text label
Widget.CATEGORY_LABEL_BOX	Text label with a border
Widget.CATEGORY_LABEL_ POPUP	Text label with a popup that blends into the background
Widget.CATEGORY_MENU_BOX	A menu box
Widget.CATEGORY_MESSAGE	For displaying a temporary warning message
Widget.CATEGORY_OPERATE	Operate buttons in T/HIS
Widget.CATEGORY_POPUP_BOX	A popup box that can contain buttons and plain text
Widget.CATEGORY_SAFE_ ACTION	Buttons (usually green) to indicate a safe action
Widget.CATEGORY_SEL_ALL	Select all
Widget.CATEGORY_TAB	Tab
Widget.CATEGORY_TABLE_ HEADER	Table (column) header
Widget.CATEGORY_TABLE_ROW	Table row
Widget.CATEGORY_TEXT_BOX	A text box
Widget.CATEGORY_TICKBOX	A tick box
Widget.CATEGORY_TITLE	Title text
Widget.CATEGORY_TOGGLE	Buttons that can be toggled, e.g. On/Off
Widget.CATEGORY_TOOL	Buttons within the tools area
Widget.CATEGORY_UNDO	Buttons which undo the last operation
Widget.CATEGORY_UNSEL_ALL	Unselect/deslect all
Widget.CATEGORY_UPDATE	Update buttons which update the screen but leave the panel open
Widget.CATEGORY_WARNING_ ACTION	Buttons (usually red) to indicate a dangerous action
Widget.NO_CATEGORY	No styling is applied. Widget colour controlled by foreground/background properties and is the same in all themes

Widget properties

Name	Type	Description
active	logical	If widget is active (true) or disabled (false)
arrows	boolean	Whether arrows will be shown for a slider (default is true). Slider Widgets only.

background	constant	Widget background colour. Can be: Widget.BLACK, Widget.WHITE, Widget.RED, Widget.GREEN, Widget.BLUE, Widget.CYAN, Widget.MAGENTA, Widget.YELLOW, Widget.DARKRED, Widget.DARKGREEN, Widget.DARKBLUE, Widget.GREY, Widget.DARKGREY, Widget.LIGHTGREY, Widget.ORANGE, Widget.DEFAULT, Widget.COLOUR_NEUTRAL, Widget.COLOUR_CONTRAST, Widget.COLOUR_CONTRAST_2, Widget.COLOUR_WARNING, Widget.COLOUR_SAFE, Widget.COLOUR_TITLE, Widget.COLOUR_INVERSE, Widget.DARKGREY_NEUTRAL, Widget.LIGHTGREY_NEUTRAL ,or a colour returned by Colour.RGB(). Note, background colours in the Window.THEME_DARK, Window.THEME_LIGHT, and Window.THEME_CLASSIC themes will be determined by the category of the widget not the background colour. To override this behaviour and use this background colour first set the widget category to Widget.NO_CATEGORY.
bottom	integer	Widget bottom coordinate
category	constant	The button category which determines the button's appearance when using the new user interface, see Window.Theme()
fontSize	integer	Widget font size in points. Currently only supports the following sizes: 6, 7, 8, 10, 12, 14, 18, 24. Can be used only with Widget.LABEL and Widget.BUTTON. Both LATIN1 and UTF-8 encoding is supported on Windows but Linux only supports LATIN1 encoding at the moment.
foreground	constant	Widget foreground colour. Can be: Widget.BLACK, Widget.WHITE, Widget.RED, Widget.GREEN, Widget.BLUE, Widget.CYAN, Widget.MAGENTA, Widget.YELLOW, Widget.DARKRED, Widget.DARKGREEN, Widget.DARKBLUE, Widget.GREY, Widget.DARKGREY, Widget.LIGHTGREY, Widget.ORANGE, Widget.DEFAULT, Widget.COLOUR_NEUTRAL, Widget.COLOUR_CONTRAST, Widget.COLOUR_CONTRAST_2, Widget.COLOUR_WARNING, Widget.COLOUR_SAFE, Widget.COLOUR_TITLE, Widget.COLOUR_LABEL, Widget.COLOUR_INVERSE, Widget.DARKGREY_NEUTRAL, Widget.LIGHTGREY_NEUTRAL ,or a colour returned by Colour.RGB(). Note, foreground colours in the Window.THEME_DARK, Window.THEME_LIGHT, and Window.THEME_CLASSIC themes will be determined by the category of the widget not the foreground colour. To override this behaviour and use this foreground colour first set the widget category to Widget.NO_CATEGORY.
hover	string	Widget hover text
imageHeight (read only)	integer	Height of widget image (pixels)
imageWidth (read only)	integer	Width of widget image (pixels)
justify	constant	Widget justification. Can be: Widget.LEFT, Widget.RIGHT or Widget.CENTRE (default).
left	integer	Widget left coordinate
lineWidth	integer	Width of lines when drawing graphics (initially 1; values 1-100 allowed).
macroTag	string	Tag to use for this widget when recording a macro. If empty then the text property value will be used.
maximum	integer	The maximum value allowed for a slider (default is 100). Slider Widgets only.
minimum	integer	The minimum value allowed for a slider (default is 0). Slider Widgets only.
monospace	boolean	true if the widget uses a monospace font instead of a proportional width font (default). Label and button Widgets only.

onChange	function	Function to call when the text in a TEXTBOX widget, the selection in a COMBOBOX widget or the value of a SLIDER is changed. The Widget object is accessible in the function using the 'this' keyword (see the example below for more details of how to define the function and how to use the 'this' keyword). To unset the function set the property to null. **Note that this function is called when the user actually types something into the textbox, selects an item in the combobox or moves the slider, NOT when the Widget.text or Widget.value property changes**.
onClick	function	Function to call when a BUTTON, LABEL, CHECKBOX or COMBOBOX widget is clicked. The Widget object is accessible in the function using the 'this' keyword (see the example below for more details of how to define the function and how to use the 'this' keyword). To unset the function set the property to null. **Note that this function is called when the user actually clicks on the button, NOT when the Widget.pushed property changes**. For the COMBOBOX widget the function is called **before** the list of items is mapped.
onPopup	function	Function to call when a BUTTON, LABEL or TEXTBOX widget is right clicked to map a popup. The Widget object is accessible in the function using the 'this' keyword. The PopupWindow can then be found by using the popupWindow property of the Widget. The function is called **before** the popup is mapped so you can change the widgets in the popup as required.
onTimer	function	Function to call for a widget when timerDelay ms have elapsed after setting this. Additionally if timerRepeat is set this function will be called repetitively, every timerDelay ms. The Widget object is accessible in the function using the 'this' keyword. To unset the function set the property to null. **Note that as soon as this property is set the timer starts!**
orientation	constant	The orientation of a slider. Can be: Widget.VERTICAL or Widget.HORIZONTAL (default). Slider Widgets only.
popupDirection	constant	How PopupWindow will be mapped relative to this widget. Can be Widget.LEFT, Widget.RIGHT, Widget.TOP or Widget.BOTTOM (default).
popupSymbol	logical	TRUE (default) if a symbol will be shown for a PopupWindow.
popupWindow	PopupWindow object	PopupWindow for this Widget. Only available for Button, Label and Textbox Widgets. To remove a PopupWindow from a Widget set to null.
pushed	logical	If widget is pushed (true) or not (false). This only affects Widget.BUTTON with the Widget.toggle property set, and Widget.CHECKBOX widgets.
right	integer	Widget right coordinate
select	constant	Selection method for ListBox Widgets. Can be: Widget.SELECT_NONE, Widget.SELECT_SINGLE or Widget.SELECT_MULTIPLE or Widget.SELECT_ENHANCED (default).
selectedItem	WidgetItem object	WidgetItem that is currently selected for a ComboBox Widget. If null no WidgetItem is selected. For a ListBox Widget this property contains the last WidgetItem that was (de)selected. To get a list of all of the selected WidgetItems use WidgetItems() to return all of the WidgetItems and inspect the WidgetItem selected property.
shown (read only)	boolean	true if the widget is visible. To alter the visibility of a widget use the Show() and Hide() methods.
step	integer	The step value of a slider (default is 1). Slider Widgets only.
text	string	Widget text. For a ComboBox Widget this will be the text for the currently selected WidgetItem
textHidden	boolean	true if the widget text is hidden and replaced by asterisks. This may be used to create textboxes to type passwords in. TextBox Widgets only.
timerDelay	integer	Delay in ms before the function set for onTimer will be called. The initial value is 1000 (ms). Also see timerRepeat.

timerRepeat	logical	If the function set for onTimer will be called once (false) or repeatedly (true). The initial value is false. Also see timerDelay.
toggle	logical	If widget can be toggled (true) or not (false). This only affects Widget.BUTTON widgets.
top	integer	Widget top coordinate
type (read only)	integer	Type of the widget. The widget type could be Widget.BUTTON, Widget.CHECKBOX, Widget.COMBOBOX, Widget.LABEL, Widget.LISTBOX, Widget.SLIDER or Widget.TEXTBOX
value	integer	The current value of a slider (initially will be the minimum value). Slider Widgets only.
window (read only)	Window object	The Window that this widget is defined in
xResolution	integer	X resolution of button when drawing lines, circles, polygons and rectangles (initially 100). X coordinates on the Widget can be from 0 (on the left of the widget) to xResolution (on the right of the widget). Available for Widget.LABEL and Widget.BUTTON Widgets.
yResolution	integer	Y resolution of button when drawing lines, circles, polygons and rectangles (initially 100). Y coordinates on the Widget can be from 0 (on the top of the widget) to yResolution (on the bottom of the widget). Available for Widget.LABEL and Widget.BUTTON Widgets.

Detailed Description

The Widget class allows you to create Widgets (buttons, textboxes etc) in a Window for a graphical user interface. Callback functions can be declared for widgets to give actions when a button is pressed or the text in a textbox is selected etc. The following example displays various widgets in a window. Several callback methods are used. The exit button allows the user to exit the script but the button is only made active if the checkbox widget is ticked. If the button widgets are pressed feedback is given to the user

```
var count = 0;
// Create window
var w = new Window("Test", 0.8, 1.0, 0.5, 0.6);
// Create all of the widgets
var l = new Widget(w, Widget.LABEL,    1, 30, 1, 7, "Text:");
var t = new Widget(w, Widget.TEXTBOX, 31, 80, 1, 7, "Enter text");
var b = new Widget(w, Widget.BUTTON,   1, 30, 8, 14, "Press me");
var b2= new Widget(w, Widget.BUTTON,  31, 61, 8, 14, "Don't press me");
var c = new Widget(w, Widget.CHECKBOX,62, 68, 8, 14);
var l2= new Widget(w, Widget.LABEL,    1, 80, 15, 21, "You haven't pressed the
button yet...");
var e = new Widget(w, Widget.BUTTON,   1, 21, 22, 28, "Exit");
// Allow button widget b2 to toggle
b2.toggle = true;
// The exit button is initially inactive
e.active = false;
// Assign the callback functions
b.onClick  = clicked;
b2.onClick = clicked;
c.onClick  = clicked;
t.onChange = changed;
e.onClick  = confirm_exit;
// Show the window and start event loop
w.Show();
////////////////////////////////////////////////////////////////////////////
function clicked()
{
// If checkbox is clicked then set the state of the exit button
    if (this === c)
    {
        Message("Checkbox clicked");
        e.active = c.pushed;
    }
// If the "Don't press me' button is pressed then change the colour if the
button is pressed in.
```

```
    else if (this === b2)
    {
        Message("I said don't press!!!");
        if (b2.pushed) b2.background = Widget.WHITE;
        else           b2.background = Widget.DEFAULT;
    }
// If the "Press me" button is pressed then update the text in the label widget
// with how many times the button has been pressed.
    else
    {
        Message("You pressed...");
        count++;
        l2.text = "Button pressed " + count + " times";
    }
}
////////////////////////////////////////////////////////////////////////////
function changed()
{
// If the user has changed the text in the textbox then give a message in
// the dialogue box
    Message("Text has changed to " + this.text);
}
////////////////////////////////////////////////////////////////////////////
function confirm_exit()
{
// Map confirm box
    var ret = Window.Question("Confirm exit", "Are you sure you want to quit?");
// If the user has answered yes then exit from the script.
    if (ret == Window.YES) Exit();
}
```

Graphics (lines, circles, rectangles etc) can be drawn on Widget.LABEL and Widget.BUTTON widgets. If these methods are used the resolution of the widget is 100 units in x and y and the origin is at the top left of the widget. See the documentation below and the WidgetItem and Window classes for more details.

Constructor

new Widget(window*[Window or PopupWindow]*, type*[constant]*, left*[integer]*, right*[integer]*, top*[integer]*, bottom*[integer]*, text (optional)*[string]*)

Description

Create a new Widget object.

Arguments

Name	Type	Description
window	Window or PopupWindow	Window or PopupWindow that widget will be created in
type	constant	Widget type. Can be Widget.LABEL, Widget.BUTTON, Widget.CHECKBOX, Widget.COMBOBOX, Widget.LISTBOX, Widget.TEXTBOX or Widget.SLIDER.
left	integer	left coordinate of widget
right	integer	right coordinate of widget
top	integer	top coordinate of widget
bottom	integer	bottom coordinate of widget
text (optional)	string	Text to show on widget (optional for LABEL, BUTTON and TEXTBOX, not required for CHECKBOX, COMBOBOX, LISTBOX and SLIDER)

Return type

Widget object

Details of functions

AddWidgetItem(item*[WidgetItem]*, position (optional)*[integer]*)

Description

Adds a WidgetItem to the Widget. Also see Widget.RemoveAllWidgetItems and Widget.RemoveWidgetItem.

Arguments

Name	Type	Description
item	WidgetItem	WidgetItem to add
position (optional)	integer	Position on Widget to add the WidgetItem. Any existing WidgetItems will be shifted down as required. If omitted the WidgetItem will be added to the end of the existing ones. **Note that positions start at 0**.

Return type

No return value

Example

To add WidgetItem wi to widget w:

```
w.AddWidgetItem(wi);
```

Circle(colour*[constant]*, fill*[boolean]*, xc*[integer]*, yc*[integer]*, radius*[integer]*)

Description

Draws a circle on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The coordinates are local to the Widget, not the Window. See properties xResolution and yResolution for more details. Note that the widget graphics will only be updated when the widget is redrawn. This is to allow the user to do multiple drawing commands on a widget. To force the widget to be redrawn call Show().

Arguments

Name	Type	Description
colour	constant	Colour of circle. See foreground for colours.
fill	boolean	If circle should be filled or not.
xc	integer	x coordinate of centre of circle.
yc	integer	y coordinate of centre of circle.
radius	integer	radius of circle.

Return type

no return value

Example

To draw a red filled circle, radius 25, at (50, 50) on widget w:

```
w.Circle(Widget.RED, true, 50, 50, 25);
```

Clear()

Description

Clears any graphics on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. Note that the widget graphics will only be updated when the widget is redrawn. This is to allow the user to do multiple drawing commands on a widget. To force the widget to be redrawn call Show().

Arguments

No arguments

Return type

no return value

Example

To clear any graphics for widget w:

```
w.Clear();
```

ClearSelection()

Description

Clears selection of any WidgetItems on the widget. Only possible for Widget.COMBOBOX and Widget.LISTBOX widgets.

Arguments

No arguments

Return type

no return value

Example

To clear selection of any WidgetItems for widget w:

```
w.ClearSelection();
```

Cross(colour (optional)*[constant]*)

Description

Draws a cross symbol on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets.

Arguments

Name	Type	Description
colour (optional)	constant	Colour of cross symbol. See foreground for colours. If omitted, current foreground colour is used.

Return type

no return value

Example

To draw a red cross symbol on widget w:

```
w.Cross(Widget.RED);
```

CtrlPressed() [static]

Description

Check to see if the Ctrl key is pressed

Arguments

No arguments

Return type

true/false

Example

To test if someone has the Ctrl key pressed:

```
if (Widget.CtrlPressed()) { ... }
```

Delete()

Description

Deletes the widget from PRIMER (removing it from the window it is defined in) and returns any memory/resources used for the widget. This function should not normally need to be called. However, sometimes a script may want to recreate widgets in a window many times and unless the old widgets are deleted PRIMER will reach the maximum number of widgets for a window (Options.max_widgets). To avoid this problem this method can be used to force PRIMER to delete and return the resources for a widget. **Do not use the Widget object after calling this method**.

Arguments

No arguments

Return type

no return value

Example

To delete widget w:

```
w.Delete();
```

DirectoryIcon(line_colour*[constant]*, fill_colour*[constant]*)

Description

Draws a directory icon on the widget. Only possible for Widget.BUTTON widgets.

Arguments

Name	Type	Description
line_colour	constant	Colour of lines of folder (only used in the old UI - in the new UI it will be ignored, a standard icon is always used). See foreground for colours.
fill_colour	constant	Colour of fill of folder (only used in the old UI - in the new UI it will be ignored, a standard icon is always used). See foreground for colours.

Return type

no return value

Example

To draw a directory icon on widget btn:

```
btn.DirectoryIcon(Widget.BLACK, Widget.YELLOW);
```

DumpImageString(filename*[string]*, format (optional)*[constant]*)

Description

Dumps a string representation of an image for a widget to a file in a form that can be used by Widget.ReadImageString(). Only possible for Widget.LABEL and Widget.BUTTON widgets.

Arguments

Name	Type	Description
filename	string	Filename to dump string representation to
format (optional)	constant	Can be Widget.RGB8 or Widget.RGB24. Before version 15 PRIMER only used 8 bits to store RGB (red, green and blue) colour information for widget images. In version 15 widget images have been changed to use 24 bits to store RGB information (8 bits for red, 8 bits for green and 8 bits for blue). Both formats are supported. If omitted the new Widget.RGB24 format will be used. See Widget.ReadImageString() for more details.

Return type

no return value

Example

To dump the image data to file 'image_data' for widget w with the old 8 bit RGB representation:

```
w.DumpImageString('image_data', Widget.RGB8);
```

To dump the image data to file 'image_data' for widget w with 24 bit RGB representation:

```
w.DumpImageString('image_data', Widget.RGB24);
```

Hide()

Description

Hides the widget on the screen

Arguments

No arguments

Return type

No return value

Example

To hide widget *w*

```
w.Hide();
```

ItemAt(index*[integer]*)

Description

Returns the WidgetItem object used at *index* in this Widget. See also Widget.TotalItems() and Widget.WidgetItems().

Arguments

Name	Type	Description
index	integer	index to return WidgetItem for. **Note that indices start at 0**.

Return type

WidgetItem object.

Example

To loop over the WidgetItems used in Widget *w*

```
for (i=0; i<w.TotalItems(); i++)
{
    wi = w.ItemAt(i);
}
```

Line(colour*[constant]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)

Description

Draws a line on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The coordinates are local to the Widget, not the Window. See properties xResolution and yResolution for more details. Note that the widget graphics will only be updated when the widget is redrawn. This is to allow the user to do multiple drawing commands on a widget. To force the widget to be redrawn call Show().

Arguments

Name	Type	Description
colour	constant	Colour of line. See foreground for colours.
x1	integer	x coordinate of start of line.
y1	integer	y coordinate of start of line.
x2	integer	x coordinate of end of line.
y2	integer	y coordinate of end of line.

Return type

no return value

Example

To draw a red line from (10, 90) to (90, 10) on widget w:

```
w.Line(Widget.RED, 10, 90, 90, 10);
```

PixelsPerUnit() [static]

Description

Returns the number of pixels per unit coordinate. This will vary depending on the monitor PRIMER is running on.

Arguments

No arguments

Return type

pixels/unit (real)

Example

To return how many pixels there are per unit coordinate:

```
var ppu = Widget.PixelsPerUnit();
```

Polygon(colour*[constant]*, fill*[boolean]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*, ... xn*[integer]*, ... yn*[integer]*)

Description

Draws a polygon on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The coordinates are local to the Widget, not the Window. See properties xResolution and yResolution for more details. Note that the widget graphics will only be updated when the widget is redrawn. This is to allow the user to do multiple drawing commands on a widget. To force the widget to be redrawn call Show().

Arguments

Name	Type	Description
colour	constant	Colour of polygon. See foreground for colours.
fill	boolean	If polygon should be filled or not.
x1	integer	x coordinate of point 1.
y1	integer	y coordinate of point 1.
x2	integer	x coordinate of point 2.
y2	integer	y coordinate of point 2.
... xn	integer	x coordinate of point n.
... yn	integer	y coordinate of point n.

Alternatively instead of x1, y1 etc you can specify a single argument which is an array of coordinates to use. In either case the number of points (x, y pairs) is limited to 500. Any extra points will be ignored.

Return type

no return value

Example

To draw a red filled triangle with corners (20, 20) and (50, 80) and (80, 20) on widget w:

```
w.Polygon(Widget.RED, true, 20, 20, 50, 80, 80, 20);
```

ReadImageFile(filename*[string]*, justify (optional)*[constant]*, transparent (optional)*[colour value (integer)]*, tolerance (optional)*[integer]*)

Description

Reads an image from a file to show on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The image will be shown on the widget underneath any text. Note that due to the way that colours are used for menus in PRIMER only a small number of colours are available for Widget images. Black and white images will display without any issues but colour images will be displayed with a reduced set of colours.

Arguments

Name	Type	Description
filename	string	Image file (BMP, GIF, JPEG or PNG) to read. To remove an image use null.

justify (optional)	constant	Widget justification. Can be a bitwise or of Widget.LEFT, Widget.RIGHT or Widget.CENTRE and Widget.TOP, Widget.MIDDLE or Widget.BOTTOM. Additionally Widget.SCALE can be used to scale the image (either reducing or enlarging it) so that it fills the widget. If omitted the default is Widget.CENTRE|Widget.MIDDLE without scaling.
transparent (optional)	colour value (integer)	Transparent colour. Must be a colour returned by Colour.RGB() in PRIMER. If given then this colour will be replaced by a transparent colour. i.e. the widget background colour will be shown. If omitted or null no transparency will be used.
tolerance (optional)	integer	Tolerance for transparent colour (0-255). Any pixels in the image that have a red, green and blue colour value within *tolerance* of the transparent colour will be transparent. For example if the transparent colour was given as Colour.RGB(255, 0, 0) and *tolerance* is 0 only pixels which have red value 255 **and** green value 0 **and** blue value 0 will be made transparent. If *tolerance* is 4, pixels which have red values between 251 and 255 **and** green values between 0 and 4 **and** blue values between 0 and 4 will be made transparent. If omitted a value of 8 will be used.

Return type

no return value

Example

To read image example.png for widget w and place it at the top left:

```
w.ReadImageFile("example.png", Widget.TOP|Widget.LEFT);
```

To read image example.png for widget w and place it at the top left, scaling it to fit the widget:

```
w.ReadImageFile("example.png", Widget.TOP|Widget.LEFT|Widget.SCALE);
```

To read image example.png for widget w and place it at the top left, replacing red with a transparent colour:

```
w.ReadImageFile("example.png", Widget.TOP|Widget.LEFT, Colour.RGB(255, 0, 0));
```

To remove an image from widget w:

```
w.ReadImageFile(null);
```

ReadImageString(string*[string]*, justify (optional)*[constant]*, transparent (optional)*[colour value (integer)]*, tolerance (optional)*[integer]*)

Description

Reads an image from a JavaScript string previously created by Widget.DumpImageString() to show on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The image will be shown on the widget underneath any text.
Note, prior to version 15 of PRIMER only a small number of colours were available for Widget images. In version 14 and earlier the RGB (red, green and blue) information for each pixel in the image was packed into a single byte (8 bits) with 3 bits for red, 3 for green and 2 for blue. Widget.DumpImageString() always returned the string beginning with "RRRGGGBB_RLE" which is this 8 bit format with run length encoding. This is format Widget.RGB8.
In version 15 support for Widget images was enhanced to give 24bit support for colours. The RGB information for each pixel has 8 bits for red, 8 bits for green and 8 bits for blue. This is format Widget.RGB24.
From version 15 Widget.DumpImageString() can either return the the old 8 bit format Widget.RGB8 (string beginning with "RRRGGGBB_RLE") or return the the new 24bit format Widget.RGB24 (string beginning with "RGB24_Z"). ReadImageString supports both formats.

Arguments

Name	**Type**	**Description**
string	string	String containing the image data previously created by Widget.DumpImageString(). To remove an image use null.

justify (optional)	constant	Widget justification. Can be a bitwise or of Widget.LEFT, Widget.RIGHT or Widget.CENTRE and Widget.TOP, Widget.MIDDLE or Widget.BOTTOM. Additionally Widget.SCALE can be used to scale the image (either reducing or enlarging it) so that it fills the widget. If omitted the default is Widget.CENTRE|Widget.MIDDLE without scaling.
transparent (optional)	colour value (integer)	Transparent colour. Must be a colour returned by Colour.RGB() in PRIMER. If given then this colour will be replaced by a transparent colour. i.e. the widget background colour will be shown. If omitted or null no transparency will be used.
tolerance (optional)	integer	Tolerance for transparent colour (0-255). Only used for the new 24bit format Widget.RGB24 (strings beginning with "RGB24_Z"). Ignored for the old 8 bit format Widget.RGB8 (strings beginning with "RRRGGGBB_RLE"). Any pixels in the image that have a red, green and blue colour value within *tolerance* of the transparent colour will be transparent. For example if the transparent colour was given as Colour.RGB(255, 0, 0) and *tolerance* is 0 only pixels which have red value 255 **and** green value 0 **and** blue value 0 will be made transparent. If *tolerance* is 4, pixels which have red values between 251 and 255 **and** green values between 0 and 4 **and** blue values between 0 and 4 will be made transparent. If omitted a value of 8 will be used.

Return type

no return value

Example

To read image data from string s for widget w and place it at the top left:

```
w.ReadImageString(s, Widget.TOP|Widget.LEFT);
```

To read image data from string s for widget w and place it at the top left, scaling it to fit the widget:

```
w.ReadImageString(s, Widget.TOP|Widget.LEFT|Widget.SCALE);
```

To read image data from string s for widget w and place it at the top left, replacing red with a transparent colour:

```
w.ReadImageString(s, Widget.TOP|Widget.LEFT, Colour.RGB(255, 0, 0));
```

To remove an image from widget w:

```
w.ReadImageString(null);
```

Rectangle(colour*[constant]*, fill*[boolean]*, x1*[integer]*, y1*[integer]*, x2*[integer]*, y2*[integer]*)

Description

Draws a rectangle on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets. The coordinates are local to the Widget, not the Window. See properties xResolution and yResolution for more details. Note that the widget graphics will only be updated when the widget is redrawn. This is to allow the user to do multiple drawing commands on a widget. To force the widget to be redrawn call Show().

Arguments

Name	Type	Description
colour	constant	Colour of rectangle. See foreground for colours.
fill	boolean	If rectangle should be filled or not.
x1	integer	x coordinate of first corner of rectangle.
y1	integer	y coordinate of first corner of rectangle.
x2	integer	x coordinate of second (opposite) corner of rectangle.
y2	integer	y coordinate of second (opposite) corner of rectangle.

Return type

no return value

Example

To draw a red filled rectangle with corners (20, 20) and (80, 80) on widget w:

```
w.Rectangle(Widget.RED, true, 20, 20, 80, 80);
```

RemoveAllWidgetItems()

Description

Removes any WidgetItems from the Widget. Also see Widget.AddWidgetItem and Widget.RemoveWidgetItem.

Arguments

No arguments

Return type

No return value

Example

To remove all WidgetItems from widget w:

```
w.RemoveAllWidgetItems();
```

RemoveWidgetItem(item[*WidgetItem*])

Description

Removes a WidgetItem from the Widget. Also see Widget.AddWidgetItem and Widget.RemoveAllWidgetItems.

Arguments

Name	Type	Description
item	WidgetItem	WidgetItem to remove

Return type

No return value

Example

To remove WidgetItem wi from widget w:

```
w.RemoveWidgetItem(wi);
```

ShiftPressed() [static]

Description

Check to see if the Shift key is pressed

Arguments

No arguments

Return type

true/false

Example

To test if someone has the Shift key pressed:

```
if (Widget.ShiftPressed()) { ... }
```

Show()

Description

Shows the widget on the screen

Arguments

No arguments

Return type

No return value

Example

To show widget w:

```
w.Show();
```

Static()

Description

Windows have two different regions for Widgets. A 'normal' region which can be scrolled if required (if the window is made smaller scrollbars will be shown which can be used to scroll the contents) and a 'static' region at the top of the Window which is fixed and does not scroll. For an example of a static region in a Window see any of the keyword editing panels. The 'Dismiss', 'Create', 'Reset' etc buttons are in the static region. By default Widgets are put into the normal region of the Window. This method puts the Widget to the static region of the Window.

Arguments

No arguments

Return type

No return value

Example

To put widget w in the static part of the window:

```
w.Static();
```

StringLength(text*[string]*, monospace (optional)*[boolean]*, fontSize (optional)*[integer]*) [static]

Description

Returns the length of a string in Widget units. This can be used to find what size a Widget must be to be able to display the string.

Arguments

Name	Type	Description
text	string	Text to find the width of
monospace (optional)	boolean	If true then width will be calculated using a monospace font. If false (default) then the normal proportional width font will be used
fontSize (optional)	integer	Calculation can be based on a defined font size, at the moment support is added only for font sizes of 6, 7, 8, 10, 12, 14, 18 and 24.

Return type

integer

Example

To get the width of string 'Example':

```
var len = Widget.StringLength('Example');
```

Tick(colour (optional)*[constant]*)

Description

Draws a tick symbol on the widget. Only possible for Widget.LABEL and Widget.BUTTON widgets.

Arguments

Name	Type	Description
colour (optional)	constant	Colour of tick symbol. See foreground for colours. If omitted, current foreground colour is used.

Return type

no return value

Example

To draw a red tick symbol on widget w:

```
w.Tick(Widget.RED);
```

TotalItems()

Description

Returns the number of the WidgetItem objects used in this Widget (or 0 if none used). See also Widget.ItemAt() and Widget.WidgetItems().

Arguments

No arguments

Return type

integer

Example

To return the total number of WidgetItems used for Widget *w*

```
var total = w.TotalItems();
```

WidgetItems()

Description

Returns an array of the WidgetItem objects used in this Widget (or null if none used). See also Widget.ItemAt() and Widget.TotalItems().

Arguments

No arguments

Return type

Array of WidgetItem objects

Example

To return WidgetItems used for Widget *w*

```
var wi = w.WidgetItems();
```

WidgetItem class

The WidgetItem class allows you to create items for combobox and listbox Widgets. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

WidgetItem properties

Name	Type	Description
background	constant	Widget background colour. Can be: Widget.BLACK, Widget.WHITE, Widget.RED, Widget.GREEN, Widget.BLUE, Widget.CYAN, Widget.MAGENTA, Widget.YELLOW, Widget.DARKRED, Widget.DARKGREEN, Widget.DARKBLUE, Widget.GREY, Widget.DARKGREY, Widget.LIGHTGREY or Widget.DEFAULT
foreground	constant	Widget foreground colour. Can be: Widget.BLACK, Widget.WHITE, Widget.RED, Widget.GREEN, Widget.BLUE, Widget.CYAN, Widget.MAGENTA, Widget.YELLOW, Widget.DARKRED, Widget.DARKGREEN, Widget.DARKBLUE, Widget.GREY, Widget.DARKGREY, Widget.LIGHTGREY or Widget.DEFAULT
hover	string	WidgetItem's hover text
index (read only)	integer	The index of this widgetitem in the parent widget (undefined if widgetitem is not assigned to a widget).
monospace	boolean	true if the widgetitem uses a monospace font instead of a proportional width font (default).
onClick	function	Function to call when a widget item in a COMBOBOX or LISTBOX widget is clicked. The Widgetitem object is accessible in the function using the 'this' keyword.
onMouseOver	function	Function to call when the mouse moves over a widget item in a COMBOBOX or LISTBOX widget. The Widgetitem object is accessible in the function using the 'this' keyword.
selectable	logical	If the widget item can be selected (true) or not (false).
selected	logical	If the widget item is selected (true) or not (false).
text	string	Widget text
widget (read only)	object	The widget that this item is defined for (null if not set)

Detailed Description

The WidgetItem class allows you to create items for combobox and listbox Widgets in a Window for a graphical user interface. The following example shows how WidgetItems are used to create a Combobox Widget and how to assign callbacks to determine when the selection has been changed.

```
var items = ["D3PLOT", "PRIMER", "SHELL", "REPORTER", "T/HIS"]
// Create window
var w = new Window("Combobox example", 0.8, 1.0, 0.5, 0.6);
// A simple combobox with a few items
var cl= new Widget(w, Widget.LABEL,   1, 30, 1, 7, "Programs:");
var cb= new Widget(w, Widget.COMBOBOX,  31, 61, 1, 7);
// Add WidgetItems to Combobox
for (i=0; i<items.length; i++)
    var wi = new WidgetItem(cb, items[i]);
// A combobox with many items showing a slider.
var li= new Widget(w, Widget.LABEL,   1, 30, 8, 14, "Long list:");
```

```
var ci= new Widget(w, Widget.COMBOBOX,  31, 61, 8, 14);
// Add WidgetItems to Combobox
// As an example we also make some of the WidgetItems unselectable and
// change the background colour
for (i=1; i<=100; i++)
{
    var wi = new WidgetItem(ci, "Item "+i);
    if ( (i % 10) == 5)
    {
        wi.selectable = false;
        wi.background = Widget.WHITE;
    }
}
var e = new Widget(w, Widget.BUTTON,   1, 21, 15, 21, "Exit");
// Assign callbacks
cb.onClick  = clicked;
cb.onChange = changed;
ci.onClick  = clicked;
ci.onChange = changed;
e.onClick   = confirm_exit
// Show the window and start event loop
w.Show();
////////////////////////////////////////////////////////////////////////////
function clicked()
{
// If combobox is clicked then print the current selection
    if (this.selectedItem)
        Message("selection is currently '"+this.selectedItem.text+"'");
}
////////////////////////////////////////////////////////////////////////////
function changed()
{
// If combobox selection is changed then print the new selection
    if (this.selectedItem)
        Message("selection is now '"+this.selectedItem.text+"'");
}
////////////////////////////////////////////////////////////////////////////
function confirm_exit()
{
// Map confirm box
    var ret = Window.Question("Confirm exit", "Are you sure you want to quit?");
// If the user has answered yes then exit from the script.
    if (ret == Window.YES) Exit();
}
```

See the documentation below and the Window and Widget classes for more details.

Constructor

new WidgetItem(widget*[Widget]*, text*[string]*, selectable (optional)*[boolean]*)

Description

Create a new WidgetItem object.

Arguments

Name	Type	Description
widget	Widget	Widget that widget item will be created in. This can be null in which case the WidgetItem will be created but not assigned to a Widget. It can be assigned later by using Widget.AddWidgetItem().
text	string	Text to show on widget item
selectable (optional)	boolean	If the widget item can be selected. If omitted the widget item will be selectable.

Return type

WidgetItem object

Window class

The Window class allows you to create windows for a graphical user interface. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Class functions

- BottomBorder()
- BuildGUIFromString(guistring*[string]*)
- Error(title*[string]*, error*[string]*, buttons (optional)*[constant]*)
- GetDirectory(initial (optional)*[string]*)
- GetFile(extension (optional)*[string]*, save (optional)*[boolean]*, initial (optional)*[string]*)
- GetFilename(title*[string]*, message*[string]*, extension (optional)*[string]*, initial (optional)*[string]*, save (optional)*[boolean]*)
- GetFiles(extension (optional)*[string]*)
- GetGraphicsWindowPosition()
- GetInteger(title*[string]*, message*[string]*, initial (optional)*[integer]*)
- GetNumber(title*[string]*, message*[string]*, initial (optional)*[real]*)
- GetPassword(title*[string]*, message*[string]*)
- GetString(title*[string]*, message*[string]*, initial (optional)*[string]*)
- Information(title*[string]*, info*[string]*, buttons (optional)*[constant]*)
- MasterResolution()
- Message(title*[string]*, message*[string]*, buttons (optional)*[constant]*)
- MiddleBorder()
- Question(title*[string]*, question*[string]*, buttons (optional)*[constant]*)
- RightBorder()
- SetGraphicsWindowPosition(left*[real]*, right*[real]*, bottom*[real]*, top*[real]*)
- SetGraphicsWindowSize(width*[integer]*, height*[integer]*)
- Theme(theme (optional)*[constant]*)
- TopBorder()
- UpdateGUI()
- Warning(title*[string]*, warning*[string]*, buttons (optional)*[constant]*)

Member functions

- Delete()
- Hide()
- Recompute()
- Redraw()
- Show(modal (optional)*[boolean]*)

Window constants

Name	Description
Window.CANCEL	Show CANCEL button
Window.NO	Show NO button
Window.NONMODAL	Allow Window.Error, Window.Question, Window.Warning etc windows to be non modal
Window.OK	Show OK button
Window.YES	Show YES button

Constants for Resizing/positioning

Name	Description
Window.BOTTOM	Bottom resizing/positioning of window
Window.CENTRE	Centre (horizontal) positioning of window
Window.LEFT	Left resizing/positioning of window
Window.MIDDLE	Middle (vertical) positioning of window
Window.REDUCE	Window is allowed to reduce in size when resizing
Window.RIGHT	Right resizing/positioning of window
Window.TOP	Top resizing/positioning of window

Constants for Themes

Name	Description
Window.THEME_CLASSIC	Use the Classic theme (Note: Not only the script will use this theme, the whole interface of the program will switch to classic)
Window.THEME_CURRENT	Use the current theme
Window.THEME_DARK	Use the Dark theme (Note: Not only the script will use this theme, the whole interface of the program will switch to dark)
Window.THEME_LIGHT	Use the Light theme (Note: Not only the script will use this theme, the whole interface of the program will switch to light)
Window.USE_OLD_UI_JS	Use the original, pre v17, theme (default). (Note:The interface of the program will NOT switch to old)

Window properties

Name	Type	Description
active	boolean	If true (default) then the window then the window is active and widgets in the window can be used. If false then the window is inactive and the widgets cannot be used.
background	constant	Window background colour. Can be: Widget.BLACK, Widget.WHITE, Widget.RED, Widget.GREEN, Widget.BLUE, Widget.CYAN, Widget.MAGENTA, Widget.YELLOW, Widget.DARKRED, Widget.DARKGREEN, Widget.DARKBLUE, Widget.GREY, Widget.DARKGREY, Widget.LIGHTGREY or Widget.DEFAULT ,or a colour returned by Colour.RGB().
bottom	real	bottom coordinate of window in range 0.0 (bottom) to 1.0 (top)
height	real	height of window
keepOnTop	boolean	If true then the window will be kept "on top" of other windows. If false (default) then the window stacking order can be changed.
left	real	left coordinate of window in range 0.0 (left) to 1.0 (right)
maxWidgets (read only)	integer	The maximum number of widgets that can be made in this window. This can be changed **before** the window is created by using Options.max_widgets. Also see totalWidgets
onAfterShow	function	Function to call **after** a Window is shown. The Window object is accessible in the function using the 'this' keyword. This may be useful to ensure that certain actions are done after the window is shown. It can also be used to show another window so this enables multiple windows to be shown. To unset the function set the property to null.
onBeforeShow	function	Function to call **before** a Window is shown. The Window object is accessible in the function using the 'this' keyword. This may be useful to ensure that buttons are shown/hidden etc before the window is shown. Note that it cannot be used to show another window. Use onAfterShow for that. To unset the function set the property to null.

onClose	function	Function to call when a Window is closed by pressing the X on the top right of the window. The Window object is accessible in the function using the 'this' keyword. To unset the function set the property to null.
resize	constant	Window resizing. By default when a Window is shown it is allowed to resize on all sides (left, right, top and bottom) to try to make enough room to show the Widgets. The behaviour can be changed by using this property. It can be any combination (bitwise OR) of Window.LEFT, Window.RIGHT, Window.TOP or Window.BOTTOM or 0. In addition Window.REDUCE can also be added to allow the window to reduce in size when resizing. Note that when Window.Show is called this property is set to 0 (i.e. not to resize on any side).
right	real	right coordinate of window in range 0.0 (left) to 1.0 (right)
showClose	boolean	If true (default) then a close (X) button will automatically be added on the top right of the window. If false then no close button will be shown.
shown (read only)	boolean	true if window is currently shown, false if not
title	string	Window title
top	real	top coordinate of window in range 0.0 (bottom) to 1.0 (top)
totalWidgets (read only)	integer	The total number of widgets that have been made in this window. This can be changed **before** the window is created by using Options.max_widgets. Also see maxWidgets
width	real	width of window

Detailed Description

The Window class allows you to make windows that you can place Widgets in to create a graphical user interface. The Widget class also gives a number of static methods for convenience. e.g. Window.GetInteger(). The following very simple example displays some text in a window with a button that unmaps the window when it is pressed and the user confirms that they want to exit.

```
// Create window with title "Text" from 0.8-1.0 in x and 0.5-0.6 in y
var w = new Window("Text", 0.8, 1.0, 0.5, 0.6);
// Create label widget
var l = new Widget(w, Widget.LABEL,   1, 40, 1,  7, "Press OK to exit");
// Create button widget
var e = new Widget(w, Widget.BUTTON, 11, 30, 8, 14, "OK");
// Assign the onClick callback method to the function confirm_exit'
e.onClick  = confirm_exit;
// Show the widget and start event loop
w.Show();
////////////////////////////////////////////////////////////////////////////
function confirm_exit()
{
// Map confirm window
    var ret = Window.Question("Confirm exit", "Are you sure you want to quit?");
// If the user has answered Yes then exit.
    if (ret == Window.YES) Exit();
}
```

See the documentation below and the Widget class for more details.

Constructor

new Window(title*[string]*, left*[real]*, right*[real]*, bottom*[real]*, top*[real]*)

Description

Create a new Window object.

Arguments

Name	Type	Description

title	string	Window title to show in title bar
left	real	left coordinate of window in range 0.0 (left) to 1.0 (right)
right	real	right coordinate of window in range 0.0 (left) to 1.0 (right)
bottom	real	bottom coordinate of window in range 0.0 (bottom) to 1.0 (top)
top	real	top coordinate of window in range 0.0 (bottom) to 1.0 (top)

Return type

Window object

Example

To create a Window 'Example' in the top right half of the screen:

```
var w = new Window('Example', 0.5, 1.0, 0.5, 1.0);
```

Details of functions

BottomBorder() [static]

Description

Returns the vertical position of the bottom border (in range 0-1). This can be used to help position windows on the screen.

Arguments

No arguments

Return type

real in range 0-1

Example

To obtain the position of the bottom border:

```
var b = Window.BottomBorder();
```

BuildGUIFromString(guistring*[string]*) [static]

Description

Builds a GUI from a JSON string created by the GUI builder.

Arguments

Name	Type	Description
guistring	string	The string to create the GUI from

Return type

object containing the GUI

Example

To create a GUI from the string json_string:

```
var gui = Window.BuildGUIFromString(json_string);
```

Delete()

Description

Deletes the window from PRIMER and returns any memory/resources used for the window. **This function should not normally need to be called**. However, in exceptional circumstances if a script recreates windows many times PRIMER may run out of USER objects on Microsoft Windows because of the way PRIMER creates and shows windows. To avoid this problem this method can be used to force PRIMER to return the resources for a window. **Do not use the Window object after calling this method**.

Arguments

No arguments

Return type

No return value

Example

To delete window w:

```
w.Delete();
```

Error(title*[string]*, error*[string]*, buttons (optional)*[constant]*) [static]

Description

Show an error message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
error	string	Error message to show in window. The maximum number of lines that can be shown is controlled by the Options.max_window_lines option.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used. By default the window will be modal. If Window.NONMODAL is also given the window will be non-modal instead.

Return type

Button pressed

Example

To show error *Critical error!\nAbort?* in window with title *Error* with Yes and No buttons:

```
var answer = Window.Error("Error", "Critical error!\nAbort?", Window.YES |
Window.NO);
if (answer == Window.YES) Exit();
```

GetDirectory(initial (optional)*[string]*) [static]

Description

Map the directory selector box native to your machine, allowing you to choose a directory. On Unix this will be a Motif selector. Windows will use the standard windows directory selector.

Arguments

Name	Type	Description

initial (optional)	string	Initial directory to start from.

Return type

directory (string), (or null if cancel pressed).

Example

To select a directory:

```
var dir = Window.GetDirectory();
```

GetFile(extension (optional)*[string]*, save (optional)*[boolean]*, initial (optional)*[string]*) [static]

Description

Map a file selector box allowing you to choose a file. See also Window.GetFiles() and Window.GetFilename().

Arguments

Name	Type	Description
extension (optional)	string	Extension to filter by.
save (optional)	boolean	If true the file selector is to be used for saving a file. If false (default) the file selector is for opening a file. Due to native operating system file selector differences, on linux new filenames can only be given when saving a file. On windows it is possible to give new filenames when opening or saving a file.
initial (optional)	string	Initial directory to start from.

Return type

filename (string), (or null if cancel pressed).

Example

To select a file using extension '.key':

```
var file = Window.GetFile(".key");
```

GetFilename(title*[string]*, message*[string]*, extension (optional)*[string]*, initial (optional)*[string]*, save (optional)*[boolean]*) [static]

Description

Map a window allowing you to input a filename (or select it using a file selector). OK and Cancel buttons are shown. See also Window.GetFile().

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
extension (optional)	string	Extension to filter by.
initial (optional)	string	Initial value.

save (optional)	boolean	If true the file selector is to be used for saving a file. If false (default) the file selector is for opening a file. Due to native operating system file selector differences, on linux new filenames can only be given when saving a file. On windows it is possible to give new filenames when opening or saving a file.

Return type

filename (string), (or null if cancel pressed).

Example

To create an file input window with title *Choose file* and message *Choose the file to open* and return the filename input:

```
var filename = Window.GetFilename("Choose file", "Choose the file to open");
```

GetFiles(extension (optional)*[string]*) [static]

Description

Map a file selector box allowing you to choose multiple files. See also Window.GetFile() and Window.GetFilename().

Arguments

Name	Type	Description
extension (optional)	string	Extension to filter by.

Return type

Array of filenames (strings), or null if cancel pressed.

Example

To select multiple files using extension '.key':

```
var files = Window.GetFiles(".key");
```

GetGraphicsWindowPosition() [static]

Description

This function returns the current position of the graphics window.

Arguments

No arguments

Return type

Array of numbers containing the left, right, bottom and top positions (in the range 0.0 to 1.0)

Example

To get the current position of the graphics window:

```
var pos = Window.GetGraphicsWindowPosition();
var l = pos[0];
var r = pos[1];
var b = pos[2];
var t = pos[3];
```

GetInteger(title*[string]*, message*[string]*, initial (optional)*[integer]*) [static]

Description

Map a window allowing you to input an integer. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
initial (optional)	integer	Initial value.

Return type

value input (integer), or null if cancel pressed.

Example

To create an input window with title *Input* and message *Input integer* and return the value input:

```
var value = Window.GetInteger("Input", "Input integer");
```

GetNumber(title*[string]*, message*[string]*, initial (optional)*[real]*) [static]

Description

Map a window allowing you to input a number. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
initial (optional)	real	Initial value.

Return type

value input (real), or null if cancel pressed.

Example

To create an input window with title *Input* and message *Input number* and return the value input:

```
var value = Window.GetNumber("Input", "Input number");
```

GetPassword(title*[string]*, message*[string]*) [static]

Description

Map a window allowing you to input a password. OK and Cancel buttons are shown.
This is identical to Window.GetString except the string is hidden and no initial value can be given.

Arguments

Name	Type	Description
title	string	Title for window.

message	string	Message to show in window.

Return type

value input (string), or null if cancel pressed.

Example

To create an input window with title *Input* and message *Input password* and return the value input:

```
var value = Window.GetPassword("Input", "Input password");
```

GetString(title*[string]*, message*[string]*, initial (optional)*[string]*) [static]

Description

Map a window allowing you to input a string. OK and Cancel buttons are shown.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window.
initial (optional)	string	Initial value.

Return type

value input (string), or null if cancel pressed.

Example

To create an input window with title *Input* and message *Input string* and return the value input:

```
var value = Window.GetString("Input", "Input string");
```

Hide()

Description

Hides (unmaps) the window.

Arguments

No arguments

Return type

No return value

Example

To hide window w:

```
w.Hide();
```

Information(title*[string]*, info*[string]*, buttons (optional)*[constant]*) [static]

Description

Show information in a window.

Arguments

Name	Type	Description
title	string	Title for window.
info	string	Information to show in window. The maximum number of lines that can be shown is controlled by the Options.max_window_lines option.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used. By default the window will be modal. If Window.NONMODAL is also given the window will be non-modal instead.

Return type

Button pressed

Example

To show information *Information* in window with title *Example* with OK and Cancel buttons:

```
var answer = Window.Information("Example", "Information", Window.OK |
Window.CANCEL);
if (answer == Window.CANCEL) Message("You pressed the Cancel button");
```

MasterResolution() [static]

Description

Returns the resolution of the master programme window in pixels

Arguments

No arguments

Return type

Array of numbers containing x and y resolution in pixels

Example

To get the resolution of the main window:

```
var res = Window.MasterResolution();
```

Message(title*[string]*, message*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
message	string	Message to show in window. The maximum number of lines that can be shown is controlled by the Options.max_window_lines option.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used By default the window will be modal. If Window.NONMODAL is also given the window will be non-modal instead.

Return type

Button pressed

Example

To show message *Press YES or NO* in window with title *Example* with YES and NO buttons:

```
var answer = Window.Message("Example", "Press YES or NO", Window.YES |
Window.NO);
if (answer == Window.NO) Message("You pressed No");
```

MiddleBorder() [static]

Description

Returns the vertical position of the middle border (in range 0-1). The middle border is the border between the tools/keywords window and the docked windows. This can be used to help position windows on the screen.

Arguments

No arguments

Return type

real in range 0-1

Example

To obtain the position of the middle border:

```
var b = Window.MiddleBorder();
```

Question(title*[string]*, question*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a question in a window.

Arguments

Name	Type	Description
title	string	Title for window.
question	string	Question to show in window. The maximum number of lines that can be shown is controlled by the Options.max_window_lines option.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted Yes and No button will be used. By default the window will be modal. If Window.NONMODAL is also given the window will be non-modal instead.

Return type

Button pressed

Example

To show question *Do you want to continue?* in window with title *Question*:

```
var answer = Window.Question("Question", "Do you want to continue?");
if (answer == Window.NO) Message("You pressed No");
```

Recompute()

Description

Recomputes the positions of widgets in the window. If you have static widgets and 'normal' widgets in a window and you show and/or hide widgets the window needs to be recomputed to refresh the graphics, scroll bars etc. Calling this method will recompute and redraw the window.

Arguments

No arguments

Return type

No return value

Example

To recompute window w:

```
w.Recompute();
```

Redraw()

Description

Redraws the window. Sometimes if you show, hide or draw graphics on widgets the window needs to be redrawn to refresh the graphics. Calling this method will redraw the window refreshing the graphics.

Arguments

No arguments

Return type

No return value

Example

To redraw window w:

```
w.Redraw();
```

RightBorder() [static]

Description

Returns the horizontal position of the right border (in range 0-1). This can be used to help position windows on the screen.

Arguments

No arguments

Return type

real in range 0-1

Example

To obtain the position of the right border:

```
var b = Window.RightBorder();
```

SetGraphicsWindowPosition(left*[real]*, right*[real]*, bottom*[real]*, top*[real]*) [static]

Description

This function allows you to move or resize the graphics window.

Arguments

Name	Type	Description
left	real	left coordinate of graphics window in range 0.0 (left) to 1.0 (right)
right	real	right coordinate of graphics window in range 0.0 (left) to 1.0 (right)
bottom	real	bottom coordinate of graphics window in range 0.0 (bottom) to 1.0 (top)
top	real	top coordinate of graphics window in range 0.0 (bottom) to 1.0 (top)

Return type

No return value

Example

To move/resize the graphics window to be in the top left half of the screen:

```
Window.SetGraphicsWindowPosition(0.0, 0.5, 0.5, 1.0);
```

SetGraphicsWindowSize(width*[integer]*, height*[integer]*) [static]

Description

This function allows you to resize the graphics window.

Arguments

Name	Type	Description
width	integer	Width of the graphics window in pixels
height	integer	Height of the graphics window in pixels

Return type

No return value

Example

To resize the graphics window to be 400 pixels wide and 300 pixels high:

```
Window.SetGraphicsWindowSize(400, 300);
```

Show(modal (optional)*[boolean]*)

Description

Shows (maps) the window and waits for user input.

Arguments

Name	Type	Description

modal (optional)	boolean	If this window is modal (true) then the user is blocked from doing anything else in PRIMER until this window is dismissed). If non-modal (false) then the user can still use other functions in PRIMER. If omitted the window will be modal. Note that making a window modal will stop interaction in all other windows and may prevent operations such as picking from working in any macros that are run from scripts.

Return type

No return value

Example

To show window w:

```
w.Show();
```

To show window w allowing the user to use other functions in PRIMER:

```
w.Show(false);
```

Theme(theme (optional)*[constant]*) [static]

Description

Set or get a user interface theme.

Arguments

Name	Type	Description
theme (optional)	constant	If it is provided it is used to set the current theme. Can be either Window.USE_OLD_UI_JS, Window.THEME_CURRENT, Window.THEME_DARK, Window.THEME_LIGHT, Window.THEME_CLASSIC.

Return type

Integer. When getting the theme one of: Window.USE_OLD_UI_JS, Window.THEME_DARK, Window.THEME_LIGHT, Window.THEME_CLASSIC

Example

To determine the current theme:

```
var ui = Window.Theme();
      if(ui == Window.THEME_DARK)
      {
          print("Theme is dark\n");
      }
      else if(ui == Window.THEME_LIGHT)
      {
          print("Theme is light\n");
      }
      else if(ui == Window.THEME_CLASSIC)
      {
          print("Theme is classic\n");
      }
      else
      {
          print("Theme is not set\n");
      }
```

To keep the original (pre v17) appearance of your JavaScript (default):

```
Window.Theme(Window.USE_OLD_UI_JS);
```

To use the current user interface theme:

```
Window.Theme(Window.THEME_CURRENT);
```

To use the dark user interface theme:

```
Window.Theme(Window.THEME_DARK);
```

TopBorder() [static]

Description

Returns the vertical position of the top border (in range 0-1). This can be used to help position windows on the screen. This is no longer used in PRIMER and will always be 1 but is left for backwards compatibility.

Arguments

No arguments

Return type

real in range 0-1

Example

To obtain the position of the top border:

```
var b = Window.TopBorder();
```

UpdateGUI() [static]

Description

Force GUI to be updated. This function is not normally needed but if you are doing a computationally expensive operation and want to update the GUI it may be necessary as the GUI update requests are cached until there is spare time to update them. Calling this function forces any outstanding requests to be flushed.

Arguments

No arguments

Return type

No return value

Example

To force update of GUI:

```
Window.UpdateGUI();
```

Warning(title*[string]*, warning*[string]*, buttons (optional)*[constant]*) [static]

Description

Show a warning message in a window.

Arguments

Name	Type	Description
title	string	Title for window.
warning	string	Warning message to show in window. The maximum number of lines that can be shown is controlled by the Options.max_window_lines option.
buttons (optional)	constant	The buttons to use. Can be bitwise OR of Window.OK, Window.CANCEL, Window.YES or Window.NO. If this is omitted an OK button will be used. By default the window will be modal. If Window.NONMODAL is also given the window will be non-modal instead.

Return type

Button pressed

Example

To show warning *Title is blank\nSet to ID?* in window with title *Warning* with Yes and No buttons:

```
var answer = Window.Warning("Warning", "Title is blank\nSet to ID?", Window.YES
| Window.NO);
if (answer == Window.NO) Message("You pressed No");
```

XlsxWorkbook class

The XlsxWorkbook class enables writing xlsx files. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- Close()

XlsxWorkbook properties

Name	Type	Description
filename (read only)	string	Name of the xlsx file

Detailed Description

The XlsxWorkbook class provides functions to enable you to create xlsx files. The following simple example shows how to write a xlsx file:

```
 var workbook = new XlsxWorkbook("C:/temp/test.xlsx");
var worksheet = new XlsxWorksheet(workbook);
worksheet.AddText(0, 0, "Hello world!");
worksheet.AddNumber(1, 0, 1.2345);
worksheet.AddImage(2, 0, "image.png");
workbook.Close();
```

Constructor

new XlsxWorkbook(filename*[string]*)

Description

Create a new XlsxWorkbook object for writing xlsx files.

Arguments

Name	Type	Description
filename	string	Filename of the xlsx file you want to write. The file will be overwritten (if it exists).

Return type

XlsxWorkbook object

Example

To create a new XlsxWorkbook object to write Xlsx file "/data/test/file.xlsx"

```
var workbook = new XlsxWorkbook("/data/test/file.xlsx");
```

Details of functions

Close()

Description

Close a Xlsx file

Arguments

No arguments

Return type

No return value

Example

To close Xlsx file workbook:

```
workbook.Close();
```

XlsxWorksheet class

Member functions

- AddImage(row*[integer]*, column*[integer]*, filename*[string]*)
- AddNumber(row*[integer]*, column*[integer]*, value*[number]*)
- AddText(row*[integer]*, column*[integer]*, text*[string]*)
- SetColumnProperties(column*[integer]*, width*[number]*)
- SetRowProperties(row*[integer]*, height*[number]*)

Constructor

new XlsxWorksheet(workbook*[XlsxWorkbook object]*, name (optional)*[string]*)

Description

Create a new XlsxWorksheet object for writing xlsx files.

Arguments

Name	Type	Description
workbook	XlsxWorkbook object	The workbook to create the worksheet in.
name (optional)	string	The name of the worksheet. If omitted the default names 'Sheet1', 'Sheet2' etc will be used.

Return type

XlsxWorksheet object

Example

To create a new worksheet in workbook

```
var worksheet = new XlsxWorksheet(workbook);
```

Details of functions

AddImage(row*[integer]*, column*[integer]*, filename*[string]*)

Description

Add an image to the Xlsx file. Note that the image will not actually be read/inserted until the workbook is written by calling XlsxWorkbook.Close so you must make sure the image file exists until then.

Arguments

Name	Type	Description
row	integer	The row in the xlsx file (rows start at zero)
column	integer	The column in the xlsx file (columns start at zero)
filename	string	Name of the image file you want to add to the xlsx file. The image can be in png or jpeg format.

Return type

No return value

Example

To add image 'C:/temp/test.png' to XlsxWorksheet worksheet on the second row, third column:

```
worksheet.AddImage(1, 2, 'C:/temp/test.png');
```

AddNumber(row*[integer]*, column*[integer]*, value*[number]*)

Description

Add number to the Xlsx file

Arguments

Name	Type	Description
row	integer	The row in the xlsx file (rows start at zero)
column	integer	The column in the xlsx file (columns start at zero)
value	number	Number you want to add to the xlsx file

Return type

No return value

Example

To add number 1.2345 to XlsxWorksheet worksheet on the second row, third column:

```
worksheet.AddNumber(1, 2, 1.2345);
```

AddText(row*[integer]*, column*[integer]*, text*[string]*)

Description

Add text to the Xlsx file

Arguments

Name	Type	Description
row	integer	The row in the xlsx file (rows start at zero)
column	integer	The column in the xlsx file (columns start at zero)
text	string	Text you want to add to the xlsx file

Return type

No return value

Example

To add text 'test' to XlsxWorksheet worksheet on the second row, third column:

```
worksheet.AddText(1, 2, 'test');
```

SetColumnProperties(column*[integer]*, width*[number]*)

Description

Set the column properties in the worksheet

Arguments

Name	Type	Description
column	integer	The column in the xlsx file (columns start at zero)
width	number	Width of the column to set

Return type

No return value

Example

To set the width of the third column in XlsxWorksheet worksheet to 30:

```
worksheet.SetColumnProperties(2, 30);
```

SetRowProperties(row*[integer]*, height*[number]*)

Description

Set the row properties in the worksheet

Arguments

Name	Type	Description
row	integer	The row in the xlsx file (rows start at zero)
height	number	Height of the row to set

Return type

No return value

Example

To set the height of the third row in XlsxWorksheet worksheet to 20:

```
worksheet.SetRowProperties(2, 20);
```

XMLParser class

The XMLParser class enables reading data from XML files. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- Parse(filename*[string]*)

XMLParser properties

Name	Type	Description
characterDataHandler	function	Function to call when character data is found. The function will be called with 1 argument which is a string containing the character data
commentHandler	function	Function to call when a comment is found. The function will be called with 1 argument which is a string containing the text inside the comment
endCDATAHandler	function	Function to call at the end of a CDATA section. The function does not have any arguments.
endElementHandler	function	Function to call when an element end tag is found. The function will be called with 1 argument which is a string containing the name of the element
startCDATAHandler	function	Function to call at the start of a CDATA section. The function does not have any arguments.
startElementHandler	function	Function to call when an element start tag is found. The function will be called with 2 arguments. Argument 1 is a string containing the name of the element. Argument 2 is an object containing the element attributes

Detailed Description

The XMLParser class provides a stream-oriented parser to enable you to read XML files. You register callback (or handler) functions with the parser and then parse the document. As the parser recognizes parts of the document, it will call the appropriate handler for that part (if you've registered one.) The document is fed to the parser in pieces. This allows you to parse really huge documents that won't fit into memory.
There are currently 6 handlers which can be set: XMLParser.startElementHandler, XMLParser.endElementHandler, XMLParser.characterDataHandler, XMLParser.commentHandler, XMLParser.startCDATAHandler and XMLParser.endCDATAHandler.
The following simple example shows how the parser could be used.

```
// Create a new parser object
var p = new XMLParser();
// assign handlers
p.startElementHandler  = startElem;
p.endElementHandler    = endElem;
p.characterDataHandler = text;
p.commentHandler       = comment;
// parse the file
p.Parse("/data/test.xml");
////////////////////////////////////////////////////////////////////////
function startElem(name, attr)
{
// handler to be called when a start element is found
// Print element name
```

```
    Println("START: " + name);
// Print attributes
    for (n in attr)
    {
        Println(" attr: " + n + "=" + attr[n]);
    }
}
function endElem(name)
{
// handler to be called when an end element is found
// Print element name
    Println("END: " + name);
}
function text(str)
{
// handler to be called when text is found
// Print text
    Println("TEXT: '" + str + "'");
}
function comment(str)
{
// handler to be called when a comment is found
// Print comment
    Println("COMMENT: '" + str + "'");
}
```

See the documentation below for more details.

Constructor

new XMLParser()

Description

Create a new XMLParser object for reading XML files.

Arguments

No arguments

Return type

XMLParser object

Example

To create a new XMLParser object

```
var p = new XMLParser();
```

Details of functions

Parse(filename*[string]*)

Description

starts parsing an XML file

Arguments

Name	Type	Description
filename	string	XML file to parse

Return type

No return value

Example

To parse XML file "/data/test.xml"

```
var p = new XMLParser();
p.Parse("/data/test.xml");
```

Xrefs class

The Xrefs class gives you access to cross references. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- GetID(type*[string]*, pos*[integer]*) [deprecated]
- GetItemID(type*[string]*, pos*[integer]*)
- GetItemType(type*[string]*, pos*[integer]*)
- GetTotal(type*[string]*)
- GetType(n*[integer]*)

Xrefs properties

Name	Type	Description
numtypes	integer	The number of different types that this item is referenced by. (read only)
total	integer	The total number of cross references of all types to this item. (read only)

Detailed Description

The Xrefs class allows you to look at what things use an item. e.g. a node may be used on several shells. See the documentation below for more details.

Details of functions

GetID(type*[string]*, pos*[integer]*) [deprecated]

This function is deprecated in version 10.0. It is only provided to keep old scripts working. We strongly advise against using it in new scripts. Support may be removed in future versions.

Description

Use Xrefs.GetItemID() instead.

Arguments

Name	Type	Description
type	string	Use Xrefs.GetItemID() instead.
pos	integer	Use Xrefs.GetItemID() instead.

Return type

No return value

GetItemID(type*[string]*, pos*[integer]*)

Description

Returns the ID of the item in the reference list.

Arguments

Name	Type	Description
type	string	The type of the item in the reference list (for a list of types see Appendix I of the PRIMER manual).
pos	integer	The position in the list for this item. **Note that positions start at 0, not 1**

Return type

ID of item

Example

To list all of the xrefs for node n:

```
var xrefs = n.Xrefs();
for (var t=0; t<xrefs.numtypes; t++)
{
    var type = xrefs.GetType(t);
    var num  = xrefs.GetTotal(type);
    for (var ref=0; ref<num; ref++)
    {
        var id = xrefs.GetItemID(type, ref);
        Message(type + " " + id + "\n");
    }
}
```

GetItemType(type*[string]*, pos*[integer]*)

Description

Returns the type of the item in the reference list. This function is only required when trying to look at cross references to *DEFINE_CURVE items. These items are used in a slightly different way in PRIMER (each time a curve is used a 'LOADCURVE REFERENCE' structure is created to store things like the units and descriptions of each axis for the curve). If you try to get the cross references for a curve all the references will be of type 'LOADCURVE REFERENCE' and numtypes will be 1. GetItemID() will correctly return the ID of the item from the 'LOADCURVE REFERENCE' structure but to get the type of the item this function is required.

Arguments

Name	Type	Description
type	string	The type of the item in the reference list (for a list of types see Appendix I of the PRIMER manual).
pos	integer	The position in the list for this item. **Note that positions start at 0, not 1**

Return type

type of item (String). For every item apart from *DEFINE_CURVE items this will be the same as the *type* argument.

Example

To list all of the xrefs for Curve c:

```
var xrefs = c.Xrefs();
for (var t=0; t<xrefs.numtypes; t++)
{
    var type = xrefs.GetType(t);
    var num  = xrefs.GetTotal(type);
    for (var ref=0; ref<num; ref++)
    {
        var id = xrefs.GetItemID(type, ref);
        var itype = xrefs.GetItemType(type, ref);
        Message(itype + " " + id + "\n");
    }
}
```

GetTotal(type*[string]*)

Description

Returns the total number of references of a type.

Arguments

Name	Type	Description
type	string	The type of the item in the reference list (for a list of types see Appendix I of the PRIMER manual).

Return type

Number of refs (integer)

Example

To find the total number of shell references that node n has:

```
var xrefs = n.Xrefs();
    var num  = xrefs.GetTotal("SHELL");
```

GetType(n*[integer]*)

Description

Returns the type for

Arguments

Name	Type	Description
n	integer	The entry in the reference types that you want the type for. **Note that entries start at 0, not 1**

Return type

The type of the item (string)

Example

To list the types of items that have cross references for node n:

```
var xrefs = n.Xrefs();
for (var t=0; t<xrefs.numtypes; t++)
{
    var type = xrefs.GetType(t);
    var num  = xrefs.GetTotal(type);
    Message(num + " references of type " + type + "\n");
}
```

Zip class

The Zip class enables reading/writing/creating zip files. More...

The PRIMER JavaScript API provides many class constants, properties and methods. For Arup to be able to extend and enhance the API in the future any constant, property or method names beginning with a lowercase or uppercase letter are reserved.
If you need to add your own properties or methods to one of the existing classes then to avoid any potential future conflict you should ensure that the name begins with either an underscore (_) or a dollar sign ($) or the name is prefixed with your own unique identifier.
For example if company 'ABC' need to add a property called 'example' then to avoid any potential future conflict use one of:

- _example
- $example
- ABC_example

Member functions

- AddFile(filename*[string]*, zipname*[string]*)
- Close()
- Information()
- Next()
- ReadFile(filename*[string]*)

Zip constants

Name	Description
Zip.APPEND	Flag to open zip file for appending
Zip.READ	Flag to open zip file for reading
Zip.WRITE	Flag to open zip file for writing

Zip properties

Name	Type	Description
filename (read only)	string	Name of the zip file
mode (read only)	constant	Mode the zip file was opened with (Zip.READ, Zip.WRITE or Zip.APPEND)

Detailed Description

The Zip class provides functions to enable you to read, write and create zip files. The following simple example shows how to write a zip file and then read it again:

```
Message("Creating zip file");
var z = new Zip("C:/temp/test.zip", Zip.WRITE);
z.AddFile("C:/temp/bpost.key",   "bpost/bpost.key");
z.AddFile("C:/temp/door.key",    "door/door.key");
z.AddFile("C:/temp/barrier.key", "other.key");
z.Close();
Message("Done");
var entry = 0;
Message("Reading zip file");
var z = new Zip("C:/temp/test.zip", Zip.READ);
while (true)
{
    entry++;
    Message("Entry "+entry);
    var info = z.Information();
    for (var x in info)
        Message("    "+x+"="+info[x]);
    z.ReadFile(entry+".txt");
    if (!z.Next()) break;
```

```
}
z.Close();
Message("Done")
```

Constructor

new Zip(filename*[string]*, mode*[constant]*)

Description

Create a new Zip object for reading/writing zip files.

Arguments

Name	Type	Description
filename	string	Filename of the zip file you want to read/write. If reading (Zip.READ) or appending (Zip.APPEND), the file must exist. If writing (Zip.WRITE) the file will be overwritten (if it exists).
mode	constant	The mode to open the file with. Can be Zip.READ, Zip.WRITE or Zip.APPEND.

Return type

Zip object

Example

To create a new Zip object to read Zip file "/data/test/file.zip"

```
var p = new Zip("/data/test/file.zip");
```

Details of functions

AddFile(filename*[string]*, zipname*[string]*)

Description

Add a file to the Zip file

Arguments

Name	Type	Description
filename	string	Name of the file you want to add to the zip file
zipname	string	Name to give the file in the zip file

Return type

No return value

Example

To add file 'C:/temp/test.key' to Zip file z with zip name 'test.key':

```
z.AddFile('C:/temp/test.key', 'test.key');
```

Close()

Description

Close a Zip file

Arguments

No arguments

Return type

No return value

Example

To close Zip file z:

```
z.Close();
```

Information()

Description

Gets information for the current entry in the Zip file such as name, size etc

Arguments

No arguments

Return type

Object with the following properties:

Name	Type	Description
compressedSize	integer	Compressed size
crc	integer	Cyclic redundancy check
name	string	Filename
uncompressedSize	integer	Uncompressed size

Example

To get the information:

```
var info = z.Information();
for (var x in info) Println(x + '=' + info[x]);
```

Next()

Description

Go to the next entry in the Zip file

Arguments

No arguments

Return type

true if there is a next entry, false if there are no more entries

Example

To go to the next entry in zip file z:

```
var next = z.Next();
```

ReadFile(filename*[string]*)

Description

Reads the current entry to a file from the Zip file

Arguments

Name	Type	Description
filename	string	Name of the file you want to create

Return type

No return value

Example

To read the current entry in Zip file z to a file ’test.key’:

```
z.ReadFile(’test.key’);
```