macromedia®
FIREWORKS®
MX

Extending Fireworks MX

Extending Fireworks MX

macromedia®

Acknowledgments

Writing: Dale Crawford, Tonya Estes, David Jacowitz, Kenneth Price

Editing: Rosana Francescato

Project management: Stuart Manning

Production: Caroline Branch, John Francis, Patrice O'Neill

Photography: Chris Basmajian

Multimedia: Aaron Begley

First Edition: June 2002

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

CONTENTS

CHAPTER 1
Extending Fireworks Overview

To extend Fireworks, you must write JavaScript code. You can use JavaScript to write your own objects, behavior actions, and commands that affect Fireworks documents and the elements within them. To accomplish these tasks, you must be proficient in JavaScript and in Fireworks.

This manual describes the Fireworks Object Model and the Fireworks JavaScript application programming interface (API)—the custom JavaScript functions that are built into Fireworks.

Prerequisites

Because Fireworks extensions must be written in JavaScript, this documentation assumes that readers are familiar with JavaScript syntax and with basic programming concepts such as functions, arguments, and data types. It also assumes that readers understand the concept of working with objects and properties. This documentation does not attempt to teach programming in general or JavaScript in particular.

Anyone who wants to extend Fireworks should have a good JavaScript reference to help with syntax questions (for example, is it `substring()` or `subString()`?). Useful JavaScript references include *JavaScript Bible* by Danny Goodman (IDG), *JavaScript: The Definitive Guide* by David Flanagan (O'Reilly), and *Pure JavaScript* by R. Allen Wyke, Jason D. Gilliam, and Charlton Ting (Sams). For a free JavaScript reference, see http://developer.netscape.com/docs/manuals/javascript.html.

Formatting nonstandard data types

In addition to the standard data types that can be passed to functions as arguments, such as integer, string, and so on, Fireworks accepts other data types for certain functions.

- Some functions take values that are Fireworks objects. These objects are explained in "The Fireworks Object Model" on page 7.
- Some functions take a string in a specific format. Others take value types that are not Fireworks objects but are JavaScript object types that are specific to Fireworks. These types of arguments are described next, in alphabetical order.

Color string

Functions that take colors as arguments use the HTML syntax of `"#rrggbb"`. You can specify a color with an alpha (transparency) component by passing a longer string of the form `"#rrggbbaa"`.

Mask

The format for mask is `{maskBounds: rectangle, maskKind: string, maskEdgeMode: string, featherAmount: int, maskData: hex-string}`.

- `maskBounds` specifies the bounding rectangle of the mask area.
- Acceptable values for `maskKind` are `"rectangle"`, `"oval"`, `"zlib compressed"`, `"rle compressed"`, or `"uncompressed"`.
- If `maskKind` is `"rectangle"` or `"oval"`, the `maskData` string is ignored, and a mask of the right shape is constructed that fills `maskBounds` and that has the edge specified by `maskEdgeMode` and `featherAmount`.
- If `maskKind` is `"zlib compressed"`, `"rle compressed"`, or `"uncompressed"`, the `maskData` string is presumed to contain 8-bit mask data in hexadecimal format that precisely matches the `maskBounds` to define the mask.

Matrix

The format for a matrix is `{matrix: [float, float, float, float, float, float, float, float, float]}`. This manual assumes that you know how to use these nine values to construct a three-by-three transformation matrix; discussion of the construction of transformation matrices is beyond the scope of this manual.

Point

The format for a point is `{x: float, y: float}`. For instance, `dom.addNewLine(startPoint, endPoint)` could look like the following example:

```
fw.getDocumentDOM().addNewLine({x:64.5, y:279.5}, {x:393.5, y:421.5});
```

Rectangle

The format for a rectangle is `{left: float, top: float, right: float, bottom: float}`. For instance, `dom.addNewOval(boundingRectangle)` could look like the following example:

```
fw.getDocumentDOM().addNewOval({left:72, top:79, right:236, bottom:228});
```

Resolution

The format for resolution is `{pixelsPerUnit: float, units: string}`. Acceptable values for units are `"inch"` or `"cm"`. For instance, `dom.setDocumentResolution(resolution)` could look like the following example:

```
fw.getDocumentDOM().setDocumentResolution({pixelsPerUnit:72, units:"inch"});
```

CHAPTER 2
The Fireworks Object Model

If you want to extend Fireworks by writing or modifying a JavaScript extensibility file, you must become familiar with the objects that Fireworks makes accessible through JavaScript. The following components comprise the Fireworks Object Model:

- Five global methods that are available from any part of the application and need not be declared as methods of a particular object. These methods are described in "Global methods" on page 9.
- Four core objects: `Document`, `Errors`, `Files`, and `Find`. These objects and their properties and methods are described in detail in "Core objects" on page 9. (The `App` object that was used in Fireworks 3 is supported for backward compatibility, but its use is deprecated in favor of the `Fireworks` object.)
- The `Fireworks` object, which is described in "The Fireworks object" on page 17.
- Numerous objects are associated with Fireworks documents, such as `ExportOptions`, `Guides`, `Path`, `Image`, and `Text`. These objects and their properties are described in "Objects within Fireworks documents" on page 20.
- A set of objects that you can use to specify the format of HTML source code when exporting from Fireworks. These are described in "HTML export objects" on page 44.

How to use the Fireworks Object Model

You send calls to the Fireworks Object Model to determine or change the current settings for a Fireworks document. For example, the following command returns the path to the Export Settings directory, which is expressed as a file://URL; `fw` references the Fireworks global object, of which `appExportSettingsDir` is a property (see "The Fireworks object" on page 17).

```
var expSetDir = fw.appExportSettingsDir;
```

Accessing a Fireworks document

All the functions listed in "Document functions" on page 58 are methods of the `Document` object, which is an object that represents a Fireworks document. To perform a function on a `Document` object, you must first get the Document Object Model (DOM) of the document. You then call the functions as methods of that DOM.

Note: You can use methods that operate on a document's DOM only on open documents.

- To use a DOM function with a document other than the active document, use the following syntax; note that `documentIndex` is a zero-based integer that specifies which document the command will affect.

  ```
  fw.documents[documentIndex].functionName();
  ```

- To use a DOM function with the active document, use `fw.getDocumentDOM().functionName()`, which is described next.

fw.getDocumentDOM()

Availability

Fireworks 3

Description

Returns the `Document` object for the active document (see "Document" on page 9). After the object is returned, you can edit its properties to make changes to the document.

Arguments

None.

Returns

A `Document` object that represents the DOM of the active document. If there is no active document, returns `null`.

Passing values

For all properties that are not read-only, you can pass values to change elements of a document. For example, the following command sets the fifth brush in the third open document to a square shape:

```
fw.documents[2].brushes[4].shape = "square";
```

The preceding example includes the following properties:

- `documents` is a property of the `Fireworks` object and contains an array of `Document` objects.
- `brushes` is a property of the `Document` object and contains an array of `Brush` objects.
- `shape` is a property of the `Brush` object.

Note: Throughout this manual, optional arguments are enclosed in {braces}.

Fireworks Object Model calls and API calls

In some cases, you can use Fireworks Object Model calls or API calls to perform the same functions. In other cases, a certain function might be available in either the Fireworks Object Model or the API, but not in both.

For example, if the first open document is the current document, the first code snippet has the same effect as the second and third code snippets. (As explained in "Accessing a Fireworks document" on page 7, `fw.getDocumentDOM()` references the current document.)

```
fw.getDocumentDOM().setDocumentResolution({pixelsPerUnit:72, units:"inch"});
fw.documents[0].resolution =72;
fw.documents[0].resolutionUnits ="inch";
```

Global methods

The following table lists the global Fireworks methods, along with their data types and, where appropriate, acceptable values and notes.

Method	Data type	Notes
`alert(`*`message`*`)`	string	Displays a string in a modal alert box, along with an OK button. Returns nothing.
`confirm(`*`message`*`)`	string	Displays a string in a modal alert box, along with OK and Cancel buttons. Returns `true` if OK is clicked, `false` if Cancel is clicked.
`prompt(`*`caption`*, *`text`*`)`	string, string	Prompts the user (with the string that is specified by text) to enter a string in a modal dialog box; the dialog box is titled with the string that is specified by caption. Returns the string entered if OK is clicked, `null` if Cancel is clicked.
`write(`*`arg1`*, *`arg2`*, ..., *`argN`*`)`	string	Same as `WRITE_HTML`. `WRITE_HTML` was created to let you differentiate HTML output calls from other JavaScript calls in your code.
`WRITE_HTML(`*`arg1`*, *`arg2`*, ..., *`argN`*`)`	string	Available only when exporting. Converts each argument to a string and writes it to the HTML output file. To enter an end-of-line character, use `"\n"`; this is converted to the correct line ending for your platform. For more information, see "HTML export objects" on page 44.

Core objects

This section describes the four core objects that are always available: `Document`, `Errors`, `Files`, and `Find`.

Note: For information on how to format nonstandard data types, such as rectangle or point, see "Formatting nonstandard data types" on page 5.

Document

The following table lists the properties and methods of the `Document` object, along with their data types and, where appropriate, acceptable values and notes. Read-only properties are marked with a bullet (•). You can also use many API calls to work with documents. For more information, see "Document functions" on page 58.

Property	Data type	Notes
`backgroundColor`	string	A color string that specifies the document canvas color (see "Color string" on page 5).
`backgroundURL`	string	Sets a general URL for a document that uses a hotspot. Everything that is not covered by the hotspot has the backgroundURL.
`brushes` •	array	Array of `Brush` objects that are available for use in the document (see "Brush" on page 21).
`currentFrameNum`	zero-based integer	The index of the current frame.
`currentLayerNum`	zero-based integer	The index of the current layer.

Property	Data type	Notes
`defaultAltText`	string	Default Alt text for the output images. It works for single and sliced images. Sliced images get the default, unless specific text is specified for a slice. Corresponds to the text that is specified in File > HTML Properties > ImageMap > AltImageDescription.
`docTitleWithoutExtension`	string	The title of the document file, without any file extension. If the document has not been saved, this string is empty.
`exportFormatOptions`	object	Identical to `exportOptions`. Included for backward compatibility with Fireworks 2.
`exportOptions`	object	`ExportOptions` object (see "ExportOptions" on page 33).
`exportSettings`	object	`ExportSettings` object (see "ExportSettings" on page 36).
`filePathForRevert`	string	The path to the file from which this document was opened, which is expressed as a file://URL, or `null` if created from scratch.
`filePathForSave`	string	The location to which this document was saved, which is expressed as a file://URL, or `null` if never saved.
`fills` •	array	Array of `Fill` objects that are available for use in the document (see "Fill" on page 38).
`frameCount`	integer	The number of frames in the current document.
`frameLoopingCount`	integer	–1 – don't repeat 0 – repeat forever > 0 – repeat this number of times
`frames` •	array	Array of `Frame` objects in the document (see "Frame" on page 38).
`gammaPreview`	Boolean	If `true`, the document should be previewed in opposite-platform gamma. If `false`, the document colors are unadjusted.
`gradients` •	array	Array of `Gradient` objects that are available for use in the document (see "Gradient" on page 39).
`gridColor`	string	A color string that specifies the color of the grid display (see "Color string" on page 5).
`gridOrigin`	point	Used to set the origin of the grid. Corresponds to the point set when dragging the ruler-origin out from the upper left of the document when rulers are visible.
`gridSize`	point	`gridSize.x` is the horizontal grid size; `gridSize.y` is the vertical grid size.
`guides` •	object	`Guides` object (see "Guides" on page 39).

Property	Data type	Notes
`height`	integer	Total height of the document, in pixels. To find the bottom edge of the document, use document. `top + document.height.`
`isDirty`	Boolean	`true` if the document was modified since the last time it was saved.
`isPaintMode` •	Boolean	`true` if the document is currently in paint-mode editing, `false` otherwise.
`isSymbolDocument` •	Boolean	`true` if the document is a Symbol or Button document, `false` if it is a normal document. You might see this when looking through the list of open documents and one is a symbol-editing window.
`isValid`	Boolean	`true` if the document is open in Fireworks; `false` otherwise. (Occasionally the JavaScript object that is associated with a document lingers after the document closes; this property lets you check for that case.)
`lastExportDirectory`	string	The path to the last directory to which the file was exported, which is expressed as a file:// URL, or `null` if the file was never exported. For instance, if the document was last exported to "`file:///files/current/logo.gif`", it returns "`file:///files/current`".
`lastExportFile`	string	The name that was used the last time the file was exported, or `null` if the file was never exported. For instance, if the document was last exported to "`file:///files/current/logo.gif`", it returns "`logo.gif`".
`layers` •	array	An array of `Layer` objects in the document (see “Layer” on page 40).
`left`	integer	Coordinate of the left edge of the document, in pixels. To find the right edge of the document, use `document.left + document.width.`
`mapType`	string	Acceptable values are "`client`", "`server`", and "`both`". Corresponds to the image-map type selected in File > HTML Properties > ImageMap.
`matteColor`	string	A color string that corresponds to the matte color specified in the Optimize panel (see “Color string” on page 5). This string is used by the `useMatteColor` property.
`onionSkinAfter`	integer	Number of frames after the current frame to show via onion skinning. Corresponds to the onion-skin controls in the left edge of the Frames panel. A value of `0` indicates no onion skinning; a very large value (such as `99,999`) indicates onion skinning of all frames after the current frame.

Property	Data type	Notes
`onionSkinBefore`	integer	Similar to `onionSkinAfter` (above), but refers to number of frames to onion skin before the current frame.
`pathAttributes`	object	`PathAttrs` object (see "PathAttrs" on page 40). This object specifies default attributes that will be applied to all newly created objects.
`pngText`	object	A structure that can be used to store various chunks of text in a well-known format. For more information, see "Using the pngText object" on page 12.
`resolution`	float	Document resolution, in pixels-per-unit (see `resolutionUnits`). The range is 1 to 5000.
`resolutionUnits`	string	The units to be used with the `resolution` property. Acceptable values are `"inch"` and `"cm"`.
`textures` •	array	Array of `Texture` objects that are available for use in the document (see "Texture" on page 32).
`top`	integer	Coordinate of the top edge of the document, in pixels. To find the bottom edge of the document, use `document.top + document.height.`
`useMatteColor`	Boolean	If `true`, the `matteColor` property is used when exporting documents with transparent backgrounds. If `false`, the `matteColor` property is ignored in this situation, and the exported file is matted against the document's canvas color.
`width`	integer	The width of the document, in pixels. To find the right edge of the document, use `document.left + document.width.`

Using the pngText object

Fireworks maintains the following fields for use with the `pngText` object:

Field name	Value
`CreationTime`	The date and time the document was created.
`Software`	The software used to create the document. Fireworks always sets this value to Macromedia Fireworks MX.

You can edit these or add your own fields, and they will be preserved across file saves.

The `pngText` object corresponds directly to the 'tEXt' chunk of the document's PNG structure.

Errors

All `Errors` properties are read-only strings that are used to make localizing scripts easier. They return localized error messages appropriate to the specific error. For example, the English version of Fireworks returns `"Memory is full."` for the `EOutOfMem` property.

The following example shows an alphabetical list of the properties of the `Errors` object:

```
EAppAlreadyRunning, EAppNotSerialized, EArrayIndexOutOfBounds,
  EBadFileContents, EBadJsVersion, EBadNesting, EBadParam, EBadParamType,
  EBadSelection, EBufferTooSmall, ECharConversionFailed, EDatabaseError,
  EDeletingLastMasterChild, EDiskFull, EDuplicateFileName, EFileIsReadOnly,
  EFileNotFound, EGenericErrorOccurred, EGroupDepth, EIllegalThreadAccess,
  EInternalError, ELowOnMem, ENoActiveDocument, ENoFilesSelected,
  ENoNestedMastersOrAliases, ENoNestedPasting, ENoSliceableElems,
  ENoSuchElement, ENotImplemented, ENotMyType, EOutOfMem, EResourceNotFound,
  ESharingViolation, EUnknownReaderFormat, EUserCanceled, EUserInterrupted,
  EWrongType
```

Files

The following table lists the methods of the `Files` object, along with their data types and, where appropriate, acceptable values and notes.

Method	Data type	Notes
`close()`	none	Closes the file referred to by this `Files` object. You are not required to call this (the file is closed when the `Files` object is destroyed), but it is useful for controlling the access to a file.
`copy(`*`docname1`*, *`docname2`*`)`	string, string	Copies the file specified in the first argument to the file specified in the second argument. Each argument must be which is expressed as a file:// URL. Only files (not directories) can be copied. The files do not need to reside on the same drive, and the method does not overwrite a file if it already exists. Returns `true` if the copy is successful; `false` otherwise.
`createDirectory(`*`dirname`*`)`	string	Creates the specified directory. Returns `true` if successful; `false` otherwise.
`createFile(`*`fileURL, {macType {,macCreator}}`*`)`	string, string, string	Creates the specified file. The file must not already exist. The first argument is the name of the file which is expressed as a file://URL. The last two optional arguments let you specify the Macintosh file type and file creator strings. If used, the *`macType`* and *`macCreator`* strings should each be strings of exactly four characters in length.
`deleteFile(`*`docOrDir`*`)`	string	Deletes the specified file or directory. Returns `true` if successful; `false` if the file or directory does not exist or cannot be deleted. Compare with `deleteFileIfExisting()`.
`deleteFileIfExisting (`*`docOrDir`*`)`	string	Deletes the specified file or directory. Returns `true` if successful; `false` if the file or directory cannot be deleted. Unlike `deleteFile()`, this method returns `true` if the file or directory does not exist.
`enumFiles(`*`docOrDir`*`)`	string	Returns an array of file URLs. If *`docOrDir`* is a directory, the array contains an entry for every file or directory that is contained in the specified directory. If *`docOrDir`* is a file, the array contains a single entry (the file passed in).
`exists(`*`docOrDir`*`)`	string	Returns `true` if *`docOrDir`* refers to a directory or file that exists; `false` otherwise.

Method	Data type	Notes
`getDirectory(`*`docname`*`)`	string	Returns only the directory name from *`docname`*, which must be which is expressed as a file://URL. For example, `Files.getDirectory("file://work/logo.png")` returns `"file:///work"`.
`getExtension(`*`docname`*`)`	string	Returns the filename extension, if any, of docname. For example, `Files.getExtension("birthday.png")` returns `".png"`. If the filename has no extension, an empty string returns. A filename that is expressed as a file://URL is acceptable.
`getFilename(`*`docname`*`)`	string	Returns just the filename from *`docname`*, which must be which is expressed as a file://URL. For example, `Files.getFilename("file:///work/logo.png")` returns `"logo.png"`.
`getLastErrorString()`	none	If the last call to a method in a `Files` object resulted in an error, returns a string that describes the error. If the last call succeeded, returns `null`.
`getTempFilePath (`*`{dirname}`*`)`	string	The argument, if used, must be expressed as a file://URL. Returns a file URL in the Temporary Files directory or in the specified directory. This method does not create a file; it simply returns a unique file URL that does not conflict with existing files in the directory. If *`dirname`* is passed and is not `null`, the URL that returns indicates a file in the specified directory rather than in the Temporary Files directory.
`isDirectory(`*`dirname`*`)`	string	The argument must be expressed as a file://URL. Returns `true` if the specified URL refers to a directory that exists; `false` otherwise.
`makePathFromDirAndFile(`*`dirname`*, *`plainFilename`*`)`	string, string	The first argument must be expressed as a file://URL. Concatenates the two arguments to return a file URL that references the specified filename in the specified directory. For example, `Files.makePathFromDirAndFile("file:///work/reports", "logo.png")` returns `"file:///work/reports/logo.png".`
`open(`*`docname`*, *`bWrite`*`)`	string, Boolean	The first argument must be expressed as a file://URL. Opens the specified file for reading or writing. If the second argument is `true`, the file opens for writing; otherwise it opens for reading. If the file cannot be opened, returns `null`; otherwise, returns a `Files` object.
`readline()`	none	Reads the next line from the file that is referred to by the current `Files` object and returns it as a string. The end-of-line character(s) are not included in the string. Returns `null` if end-of-file is reached or if the line is more than 2048 characters.
`rename(`*`docname`*, *`newPlainFilename`*`)`	string, string	*`docname`* is a file path or a file URL to the file that you want to rename. *`newPlainFilename`* is the new name to assign to the file. The `rename` method returns a URL path of the newly renamed file if successful; otherwise Fireworks returns `null`.

Method	Data type	Notes
`setFilename(`*`docname`*`, `*`newPlainFilename`*`)`	string, string	The first argument must be expressed as a file:// URL. Returns a file URL with *`docname`* replaced by *`newPlainFilename`*. For example, `Files.setFilename("file:///work/logo.png", "oldlogo.png")` returns `"file:///work/oldlogo.png"`. This method does not affect the file on disk; it simply provides a convenient way to manipulate file URLs. To change the name on disk, use `rename()`.
`swap(`*`docname1`*`, `*`docname2`*`)`	string, string	Each argument must be expressed as a file://URL. Swaps the contents of the two specified files, so that each file contains the contents of the other file. Only files (not directories) can be swapped, and both files must reside on the same drive. Returns `true` if the swap is successful; `false` otherwise.
`write(`*`textString`*`)`	string	Writes the specified string to the file that is referred to by the current `Files` object. No end-of-line characters are appended; to include one, use `"\n"`.

Find

There are several ways to specify a `Find` object, depending on what you want to find and replace. Use the `whatToFind` property to specify the type of find operation, along with the properties that are associated with each legal value for `whatToFind`. These properties are listed in the following tables. Read-only properties are marked with a bullet (•).

To find and replace text

Property	Data type	Notes
`whatToFind`	string	`"text"`
`find`	string	Text to find.
`matchCase`	Boolean	If `true`, the search is case-sensitive. Defaults to `false`.
`regExp`	Boolean	If `true`, the `find` and `replace` text is interpreted as a Regular Expression. Defaults to `false`.
`replace`	string	Text to use as replacement text.
`wholeWord`	Boolean	If `true`, only whole words matching the search text are found. Defaults to `false`.

To find and replace fonts and styles

Property	Data type	Notes
`whatToFind`	string	`"font"`
`find`	string	Name of font to find.
`replace`	string	Name of font to use as replacement.

`findStyle`	integer	Number that represents the style to find:
		`AnyStyle = -1`
		`Plain = 0`
		`Bold = 1`
		`Italic = 2`
		`BoldItalic = 3`
		`Underline = 4`
		`BoldUnderline = 5`
		`ItalicUnderline = 6`
		`BoldItalicUnderline = 7`

`replaceStyle`	integer	Number that represents the style to be used as replacement.
`findMinSize`	integer	0 to 9999
`findMaxSize`	integer	0 to 9999
`replaceSize`	integer	0 to 9999, or pass `-1` to leave size as is

To find and replace colors, fills, strokes, and effects

Property	Data type	Notes
`whatToFind`	string	`"color"`
`find`	string	A color string that specifies the color to find (see "Color string" on page 5).
`replace`	string	A color string that specifies the color to use as a replacement (see "Color string" on page 5).
`fills`	Boolean	If `true`, fills that match the specified colors are replaced.
`strokes`	Boolean	If `true`, strokes that match the specified colors are replaced.
`effects`	Boolean	If `true`, effects that match the specified colors are replaced.

To find and replace URLs

Property	Data type	Notes
`whatToFind`	string	`"url"`
`find`	string	URL to find, which is expressed as a file://URL.
`replace`	string	URL to use as replacement text, which is expressed as a file://URL.
`wholeWord`	Boolean	If `true`, only whole words that match the search text are found. Defaults to `false`.

Property	Data type	Notes
`matchCase`	Boolean	If `true`, the search is case-sensitive. Defaults to `false`.
`regExp`	Boolean	If `true`, the `find` and `replace` text is interpreted as a Regular Expression. Defaults to `false`.

To find and replace nonwebsafe colors with the closest websafe color

Property	Data type	Notes
`whatToFind`	string	`"nonwebcolor"`
`effects`	Boolean	If `true`, colors in effects are replaced. Default value is `false`.
`fills`	Boolean	If `true`, colors in fills are replaced. Default value is `false`.
`strokes`	Boolean	If `true`, colors in strokes are replaced. Default value is `false`.

The Fireworks object

The `Fireworks` object is the global object, which you can use to set or retrieve properties that relate to the current operating environment. (The `App` object that was used in Fireworks 3 is supported for backward compatibility, but its use is deprecated in favor of the `Fireworks` object.)

The following table lists the properties and methods of the `Fireworks` object, along with their data types and, where appropriate, acceptable values and notes. Read-only properties are marked with a bullet (•).

Note: For information on how to format nonstandard data types, such as rectangle or point, see “Formatting nonstandard data types” on page 5.

Refer to the Fireworks object by using `fw.propertyName` or `fireworks.propertyName`. Note that `fireworks` must be lowercase.

Property or Method	Data type	Notes
`activeViewScale`	float	The scaling (zoom value) of the active view. 1.0=100% of the normal view.
`appBatchCodeDir` •	string	The path to the Batch Code directory, which is expressed as a file://URL.
`appDir` •	string	The path to the directory that contains the Fireworks application, which is expressed as a file://URL.
`appExportSettingsDir` •	string	The path to the Export Settings directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.

Property or Method	Data type	Notes
`appFavoritesDir` •	string	The path to the URL Libraries directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.
`appHelpDir` •	string	The path to the directory that contains the Fireworks help file, which is expressed as a file://URL.
`appHtmlCodeDir` •	string	The path to the HTML Code directory, which is expressed as a file://URL.
`appJsCommandsDir` •	string	The path to the Commands directory, which is expressed as a file://URL.
`appJsExtensionsDir` •	string	The path to the JSExtensions directory, which is expressed as a file://URL.
`appMacCreator` •	string	`"MKBY"`
`appMacJsfFileType` •	string	`"TEXT"`
`appName` •	string	The name of the application (`"Fireworks MX"`). This attribute is part of the common API, so it also appears as `app.appName` (as implemented in Dreamweaver).
`appPatternsDir` •	string	The path to the Patterns directory, which is expressed as a file://URL.
`appPrefsDir`	string	The path to the Preferences directory, which is expressed as a file://URL.
`appPresetsDir` •	string	The path to the Presets directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.
`appSettingsDir` •	string	The path to the Settings directory, which is expressed as a file://URL.
`appStylesDir` •	string	The path to the Styles directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.
`appSwfCommandsDir`	string	The path to the SWF Commands directory, which is expressed as a file://URL.
`appSymbolLibrariesDir` •	string	The path to the Libraries directory, which is expressed as a file://URL.
`appTexturesDir` •	string	The path to the Textures directory, which is expressed as a file://URL.
`appXtrasDir` •	string	The path to the Xtras directory, which is expressed as a file://URL.

Property or Method	Data type	Notes
`batchStatusString`	string	The string that currently appears in the Batch Progress dialog box. Set this property to change the string being displayed. Use with `progressCountCurrent` and `progressCountTotal`.
`currentScriptDir`	string	The path to the directory of the currently running script, which is expressed as a file://URL (or could be `null`). This path goes to the directory in which the script resides, not a full file path to the script itself (it excludes the script's filename).
`currentScriptFileName`	string	The filename of the currently running script (or could be `null`). This name is the script's filename, not the full path.
`documentList` •	array	Array of the current open `Document` objects (see "Document" on page 9). If no document is open, it returns an array of length zero.
`documents` •	array	Array of the current open `Document` objects (see "Document" on page 9). If no document is open, returns an array of length zero.
`historyPalette` •	object	`History panel` object. There are no DOM properties for the History panel, only API calls. For more information, see "History panel functions" on page 197.
`platform` •	string	The string `"mac"` if Fireworks is running on the Macintosh, or `"win"` if running on Windows.
`progressCountCurrent`	integer	The first number (*x*) that appears in the Batch Progress dialog box, in the "File x of y" field. Set this property to change the number.
`progressCountTotal`	integer	The second number (*y*) that appears in the Batch Progress dialog box, in the "File x of y" field. Set this property to change the number.
`screenRect` •	rectangle	The size of the main screen on this computer, in pixels. Useful for positioning windows or panels.
`selection`	array	Array of the selected objects in the active document. If nothing is selected, it returns an array of length zero. If no document is open, it returns `null`.
`selectedMask`	object	If a single item is selected and that item is a mask, this property returns an `ElementMask` (see "ElementMask" on page 32); otherwise it returns `null`.
`styles` •	array	Array of the `Style` object that is currently loaded in the Style panel (see "Style" on page 41).
`textOutputEncoding`	string	The default text encoding for any text file that the JavaScript interpreter generates. Use `"iso-8859-1"` for ASCII or `"utf-8"` for Unicode.

Property or Method	Data type	Notes
`userJsCommandsDir`	string	The path to the user-level Commands directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.
`userSwfCommandsDir`	string	The path to the user-level SWF Commands directory, which is expressed as a file://URL. In Fireworks MX, this folder is stored on a per-user basis on multiuser systems. Even on single-user systems, this folder is not inside the Fireworks installation directory.
`xhtmlFormat`	Boolean	Determines whether the JavaScript interpreter should output XHTML formatted files or HTML formatted files; XHTML (`true`) or HTML (`false`).

Using fw.locateDocDialog()

The *`formatArray`* argument of the `locateDocDialog()` method is an array of strings such as the ones shown in the following example:

```
["formatname1","formatname2","formatname3",…"formatnameN"]
```

The following table lists acceptable values for `formatname` and the file type each value represents.

Value	File type
`"ADOBE AI3"`	Adobe Illustrator
`"Fireworks JavaScript"`	Fireworks JSF
`"kMoaCfFormat_BMP"`	bitmap
`"kMoaCfFormat_FreeHand7and8"`	Macromedia FreeHand 7 or 8
`"kMoaCfFormat_GIF"`	GIF
`"kMoaCfFormat_JPEG"`	JPEG
`"kMoaCfFormat_PICT"`	Macintosh PICT
`"kMoaCfFormat_RTF"`	Rich Text
`"kMoaCfFormat_Text"`	Plain text
`"kMoaCfFormat_TIFF"`	TIFF
`"PNG"`	PNG
`"PS30"`	Photoshop PSD

Objects within Fireworks documents

This section describes the objects that provide access to elements within a Fireworks document. For syntax on accessing Fireworks documents and elements within them, see “Accessing a Fireworks document” on page 7 and “Passing values” on page 8.

Note: For information on how to format nonstandard data types, such as rectangle or point, see “Formatting nonstandard data types” on page 5.

Behavior

The following table lists the properties of the `Behavior` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`call`	string	The JavaScript call for the behavior. For legal values, see "Using the addBehavior() function" on page 201.
`event`	string	Acceptable values are `"onMouseOver"`, `"onClick"`, `"onMouseOut"`, `"onLoad"`, and `"**ANY**"` (the `**ANY**` argument is used as a wildcard value in some situations).

Brush

The following table lists the properties of the `Brush` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`alphaRemap`	string	Acceptable values are `"none"`, `"white neon"`, `"harsh wet"`, `"smooth neon"`, `"wavy gravy"`, and `"white neon edge"`.
`angle`	integer	0 to 360
`antiAliased`	Boolean	If `true`, the brush edges are anti-aliased.
`aspect`	float	0 to 100
`blackness`	float	0 to 100
`category`	string	Determines in which subsection of the Stroke panel the brush will appear (for example, Pencil, Airbrush, and so on).
`concentration`	float	0 to 100
`diameter`	integer	0 to 1000
`feedback`	string	Acceptable values are `"none"`, `"brush"`, and `"background"`.
`flowRate`	float	0 to 100
`maxCount`	integer	0 to 64
`minSize`	float	0 to 100
`name`	string	The name of the brush, which is visible in the Stroke panel.
`sensitivity_x_y`	integer	0 to 100, where *x* is a value of `pressure, speed, hDir, vDir, random`; and *y* is a value of: `size, angle, opacity, blackness, scatter, hue, lightness, saturation`. For example, `sensitivity_pressure_size`.
`shape`	string	Acceptable values are `"circle"` and `"square"`.
`softenMode`	string	Acceptable values are `"bell curve"` and `"linear"`.
`softness`	float	0 to 100

Property	Data type	Notes
`spacing`	float	0 to 500 (a percentage, as much as 500 percent)
`textureBlend`	float	0 to 100
`textureEdge`	float	0 to 100
`tipColoring`	string	Acceptable values are `"random"`, `"uniform"`, `"complementary"`, `"hue"`, and `"shadow"`.
`tipCount`	integer	1 to 32
`tipSpacing`	float	0 to 100
`tipSpacingMode`	string	Acceptable values are `"random"`, `"diagonal"`, and `"circular"`.
`type`	string	Acceptable values are `"natural"` and `"simple"`.

Contour

The following table lists the properties of the `Contour` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`isClosed`	Boolean	If `true`, the path is closed by connecting the final point in the contour with the first point.
`nodes`	array	Array of `ContourNode` objects on the contour (see `ContourNode`).

ContourNode

The following table lists the properties of the `ContourNode` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`dynamicInfo`	array	Array of `ContourNodeDynamicInfo` objects on this `ContourNode` object (see “ContourNodeDynamicInfo” on page 23).
`isCurvePoint`	Boolean	If `true`, this point’s control points are constrained to be linear with the main point, which forces a smooth curve. If `false`, there are no constraints on the control points.
`isSelectedPoint`	Boolean	If `true`, this point was subselected (for example, by the subselection tool).
`predX`	float	The x coordinate of the contour node’s preceding control point.
`predY`	float	The y coordinate of the contour node’s preceding control point.
`randomSeed`	integer	0 to 65,535
`succX`	float	The x coordinate of the contour node’s following control point.
`succY`	float	The y coordinate of the contour node’s following control point.

Property	Data type	Notes
`x`	float	The *x* coordinate of the contour node's main control point.
`y`	float	The *y* coordinate of the contour node's main control point.

ContourNodeDynamicInfo

The following table lists the properties of the `ContourNodeDynamicInfo` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`duration`	float	0.0 to 65,535.0 milliseconds
`pressure`	float	0.0 to 1.0
`velocity`	float	0.0 to 255.9999 pixels-per-millisecond

Effect

Each `Effect` object has a different set of properties because every effect has different attributes that can be set. The properties for various `Effect` objects are listed in the following tables, in alphabetical order.

Note: In addition to the listed properties, each `Effect` object has two optional string properties: `category` and `name`.

Bevel

Use the `BevelType` property of this effect to set a bevel as inner, outer, raised embossed, inset embossed, or glow effect.

Property	Data type	Notes
`AngleSoftness`	integer	Specifies the blur, or feather amount, for the shadow and highlight colors of the bevel.
`BevelContrast`	integer	0 to 100 percent

`BevelType`	integer	`InnerBevel = 0`
		`OuterBevel = 1`
		`RaiseEmboss = 2`
		`InsetEmboss = 3`
		`GlowEffect = 4`

`BevelWidth`	integer	The width of the bevel, in pixels.

`ButtonState`	integer	`BevelButtonUp = 0`
		`BevelButtonOver = 1`
		`BevelButtonDown = 2`
		`BevelButtonHit = 3`

`DownBlendColor`	string	A color string that specifies the color that is blended on top of the image if `ButtonState = 2` (BevelButtonDown) (see "Color string" on page 5).
`EdgeThreshold`	integer	Controls the opacity at which the edge of the effect is defined. Use `1` if `BevelType = 4` (for GlowEffect); otherwise, use `0`.

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{7fe61102-6ce2-11d1-8c76000502701850}"`
`EmbossFaceColor`	string	A color string that specifies the color that is blended onto the face of the object when embossing (see “Color string” on page 5).
`GlowStartDistance`	integer	Specifies how far away from the object the glow starts, in pixels. Specify a negative value to create “ring” type glows and a positive value to create “halo” type glows.
`GlowWidth`	integer	The width of the glow, in pixels.
`HiliteColor`	string	A color string that specifies the color that is blended to provide the spectral lighting type effect (see “Color string” on page 5). Used by beveling only. Currently white is always used for internally created effects (although any value should work). This is the complement of `ShadowColor`.
`HitBlendColor`	string	A color string that specifies the color that is blended on the face of the image if `ButtonState = 3` (BevelButtonHit) (see “Color string” on page 5).
`LightAngle`	integer	The light angle, in degrees, that is used to create the light and shadow effects for the bevel.
`MaskSoftness`	integer	The feather amount on the glow edge, in pixels.
`OuterBevelColor`	string	A color string that specifies the color of the outer bevel effect (see “Color string” on page 5).
`ShadowColor`	string	A color string that specifies the color that is blended to provide the bevel shadow effect (see “Color string” on page 5). Currently black is always used for internally created effects (though any value should work). This is the complement of `HiliteColor`.
`ShowObject`	Boolean	Default value is `false`.
`SlopeMultiplier`	float	A multiplier that is used to calculate the magnitude of the bevel slope. Default effects all use 1, but other values should work. For example, 0.5 gives a more subtle slope and 2.0 gives a sharper slope.

`SlopeType`	integer	`flat slope = 0`
		`smooth slope = 1`
		`inverted smooth slope = 2`
		`frame 1 slope = 3`
		`frame 2 slope = 4`
		`ring slope = 5`
		`ruffle slope = 6`

Blur

Property	Data type	Notes
`EffectMoaID`	string	`"{f1cfce41-718e-11d1-8c8200a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.

Blur More

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{f1cfce42-718e-11d1-8c8200a024cdc039}"`

Brightness/Contrast

Property	Data type	Notes
`brightness_amount`	integer	-100 to 100
`contrast_amount`	integer	-100 to 100
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{3439b08c-1921-11d3-9bde00e02910d580}"`

Convert to Alpha

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{2932d5a2-ca48-11d1-8561000502701850}"`

Curves

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{3439b08e-1923-11d3"-9bde00e02910d580}"`

`rgb_points`
`red_points`
`green_points`
`blue_points`
	vector of points	Each of these properties is a vector of points where x = input level and y = output level. All x and y values must be between 0 and 255, and the points must be sorted in ascending order of x coordinate.

Drop Shadow

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{a7944db8-6ce2-11d1-8c76000502701850}"`
`ShadowAngle`	float	The angle of the shadow, in degrees.
`ShadowBlur`	integer	The feathering amount of the shadow edges, in pixels.
`ShadowColor`	string	A color string that specifies the color of the shadow (see “Color string” on page 5).
`ShadowDistance`	integer	The offset of the shadow, in pixels.

`ShadowType`	integer	`0 = normal shadow`
		`1 = knockout shadow`

Find Edges

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{fc7093f1-f95c-11d0-8be200a024cdc039}"`

Gaussian Blur

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{d04ef8c0-71b3-11d1-8c8200a024cdc039}"`
`gaussian_blur_radius`	float	0.1 to 250

Hue/Saturation

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{3439b08d-1922-11d3-9bde00e02910d580}"`
`hue_amount`	integer	-180 to 180 if `hls_colorize` is `false`; 0 to 360 if `hls_colorize` is `true`.
`saturation_amount`	integer	-100 to 100 if `hls_colorize` is `false`; 0 to 100 if `hls_colorize` is `true`.
`lightness_amount`	integer	0 to 100
`hls_colorize`	Boolean	Specifies whether the effect should automatically colorize. Default value is `false`.

Inner Shadow

Property	Data type	Notes
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`EffectMoaID`	string	`"{5600f702-774c-11d3-baad0000861f4d01}"`
`ShadowAngle`	integer	The angle of the shadow, in degrees.
`ShadowBlur`	integer	The feathering amount of the shadow edges, in pixels.
`ShadowColor`	string	A color string that specifies the color of the shadow (see "Color string" on page 5).
`ShadowDistance`	integer	The offset of the shadow, in pixels.

`ShadowType`	integer	`0 = normal shadow`
		`1 = knockout shadow`

Invert

Property	Data type	Notes
`EffectMoaID`	string	`"{d2541291-70d6-11d1-8c8000a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.

Levels

Property	Data type	Notes
`EffectMoaID`	string	`"{d04ef8c1-71b4-11d1-8c8200a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.

`source_low_rgb`
`source_high_rgb`
`source_low_red`
`source_high_red`
`source_low_green`
`source_high_green`
`source_low_blue`
`source_high_blue`
	integer	These values are all input levels to the filter, with values of 0 to 255.

Property	Data type	Notes
`dest_low_rgb`	integer	These values are all output levels to the filter, with values of 0 to 255.
`dest_high_rgb`		
`dest_low_red`		
`dest_high_red`		
`dest_low_green`		
`dest_high_green`		
`dest_low_blue`		
`dest_high_blue`		
`gamma_rgb`	float	These values are all gamma levels to the filter, with values of 0.1 to 10.0.
`gamma_red`		
`gamma_green`		
`gamma_blue`		

Sharpen

Property	Data type	Notes
`EffectMoaID`	string	`"{c20952b1-fc76-11d0-8be700a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.

Sharpen More

Property	Data type	Notes
`EffectMoaID`	string	`"{1f2f2591-9db7-11d1-8cad00a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.

Unsharp Mask

Property	Data type	Notes
`EffectMoaID`	string	`"{f1cfce44-718e-11d1-8c8200a024cdc039}"`
`EffectIsVisible`	Boolean	If `false`, the effect is included but temporarily hidden. Default value is `true`.
`unsharp_mask_amount`	integer	1 to 500
`unsharp_mask_radius`	float	0.1 to 250
`unsharp_mask_threshold`	integer	0 to 255

EffectList

The following table lists the properties of the `EffectList` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`category`	string	Specifies which subheading in the Effects panel to use.
`effects`	array	Array of `Effect` objects (see “Effect” on page 23).
`name`	string	The name that appears in the Effects panel.

Element

`Element` is an abstract or base class; nothing of class `Element` ever exists. However, it is useful for simplifying the other class descriptions. Read-only properties are marked with a bullet (•).

Property	Data type	Notes
`blendMode`	string	Acceptable values are `"normal"`, `"multiply"`, `"screen"`, `"darken"`, `"lighten"`, `"difference"`, `"hue"`, `"saturation"`, `"color"`, `"luminosity"`, `"invert"`, `"tint"`, and `"erase"`.
`effectList`	object	`EffectList` object (see “EffectList” on page 29).
`height` •	float	Read-only in the base class; other properties or API calls are used to resize specific types of elements.
`left`	float	Can round to an integer.
`mask`	object	`ElementMask` object (see “ElementMask” on page 32). Returns `null` if the element has no element mask.
`name`	string	Can be `null` (removes any existing name).
`opacity`	float	Acceptable values, 0 to 100, represent percent opacity.
`top`	float	Can round to an integer.
`visible`	Boolean	If `false`, the element is hidden. Default value is `true`.
`width` •	float	Read-only in the base class; other properties or API calls are used to resize specific types of elements.

Group

`Group` is a subclass of the base class `Element` and contains the following properties in addition to those in `Element` (see “Element” on page 29).

Property	Data type	Notes
`elements`	array	Array of `Element` objects in the group (see “Element” on page 29).
`groupType`	string	Acceptable value is `"normal"`. (`"mask to image"` and `"mask to path"` are deprecated in Fireworks MX.)

Image

`Image` is a subclass of the base class `Element` (see “Element” on page 29). It contains no properties or methods other than those in `Element`.

Instance

`Instance` is a subclass of the base class `Element` and contains the following properties in addition to those in `Element` (see “Element” on page 29). Read-only properties are marked with a bullet (•).

Property	Data type	Notes
`altText`	string	The alternate text description.
`instanceType` •	string	The type of Element, for example `"graphic"`, `"button"`, or `"animation"`.
`symbolID` •	string	An arbitrary string that uniquely identifies the symbol that owns this instance.
`targetText`	string	The target.
`transformMode`	string	Acceptable values are `"paths"` and `"pixels"`.
`urlText`	string	The link text.

Hotspot

A `Hotspot` generates an image map during HTML export. `Hotspot` is a subclass of the base class `Element` and contains the following properties in addition to those in `Element` (see “Element” on page 29).

Property	Data type	Notes
`altText`	string	Text that is written into the HTML Alt tag when exporting.
`behaviors`	array	Array of `Behavior` objects for the hotspot “Behavior” on page 21.
`color`	string	Color in which the hotspot is drawn in the document window. Default value is `"#00FFFF"`.
`contour`	object	`Contour` object for the hotspot “Contour” on page 22. Used only if `shape="polyline"`; otherwise `null`.
`shape`	string	Acceptable values are `"rectangle", "circle"`, and `"polyline"`.
`targetText`	string	Text that is written into the HTML Target tag when exporting.
`urlText`	string	Text that is written into the HTML Href tag when exporting.

SliceHotspot

A `SliceHotspot` generates an image slice during HTML export. `SliceHotspot` is a subclass of the base class `Hotspot` and contains the following properties in addition to those in `Hotspot` (see "Hotspot" on page 30). Read-only properties are marked with a bullet (•).

Property	Data type	Notes
`baseName`	string	Base name for slice filenames, or `null` for automatic name.
`exportOptions`	object	`ExportOptions` object (see "ExportOptions" on page 33); `null` if using current document defaults.
`htmlText`	string	If `sliceKind` is `"empty"`, this text is exported instead of the image. The default is an empty string.
`sliceID` •	string	An arbitrary string that uniquely identifies this slice.
`sliceKind`	string	`"image"` generates an image; `"empty"` generates the text that is specified by `htmlText`.
`tdTagText`	string	This string contains all the attributes of a table cell except the `colspan` and `rowspan` values. An example value is `"bgcolor=ff0000" valign="top""`.

Path

Path is a subclass of the base class `Element` and contains the following properties in addition to those in `Element` (see "Element" on page 29).

Property	Data type	Notes
`contours`	array	Array of `Contour` objects on this `Path` object (see "Contour" on page 22).
`pathAttributes`	object	`PathAttrs` object (see "PathAttrs" on page 40).
`randSeed`	float	A 32-bit integer. JavaScript integers hold only 31-bit numbers, so it is stored as a floating-point number.
`textureOffset`	point	If the path has a textured brush or fill, specifies the offset of the texture's origin.

Text

`Text` is a subclass of the base class `Element` and contains the following properties in addition to those in `Element` (see "Element" on page 29).

Property	Data type	Notes
`antiAliased`	Boolean	If `true` (the default), it anti-aliases the text.
`antiAliasMode`	string	Acceptable values are `"smooth"`, `"crisp"`, and `"strong"`. This value is ignored if the `antiAliased` property is `false`.
`autoKern`	Boolean	If `true`, uses pair-kerning information in the font(s) to kern the text. If `false`, pair-kerning information in the font(s) is ignored. Default value is `true`.

Property	Data type	Notes
`orientation`	string	Acceptable values are `"horizontal left to right"` (the default), `"vertical right to left"`, `"horizontal right to left"`, and `"vertical left to right"`.
`pathAttributes`	object	`PathAttrs` object (see "PathAttrs" on page 40).
`randSeed`	float	A 32-bit integer. JavaScript integers hold only 31-bit numbers, so it is stored as a floating-point number.
`textRuns`	object	`TextRuns` object (see "TextRuns" on page 44).
`textureOffset`	point	If the text has a textured brush or fill, specifies the offset of the texture's origin.
`transformMode`	string	Acceptable values are `"paths"` and `"pixels"`.

Texture

`Texture` is a subclass of the base class `Element` and contains the following read-only property in addition to those in `Element` (see "Element" on page 29).

Property (read-only)	Data type	Notes
`name`	string	The name that appears in the Brush or Fill panels.

ElementMask

The following table lists the properties of the `ElementMask` object, which is new in Fireworks 4, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`autoExpandImages`	Boolean	If `true`, and the element mask is an image, the image is always automatically expanded to fill the entire document, with areas "outside" the image showing through. If `false` (or if the element mask is not an image), areas "outside" the element mask are knocked out.
`element`	object	`Element` object (see "ElementMask" on page 32).
`enabled`	Boolean	If `true`, the mask applies to the element. If `false`, the mask remains present but does not visually affect the element in any way. Default value is `true`.
`linked`	Boolean	If `true`, moving the mask moves the element that owns it, and vice versa. If `false`, moving the mask does not affect the element that owns it (and moving the element does not affect the mask). Default value is `true`.
`mode`	string	Acceptable values are `"mask to image"` and `"mask to path"`.
`owner`	object	The element (image, path, text, and so on) that owns the mask.
`showAttrs`	Boolean	If `true`, and `mode` is `"mask to path"`, the mask element's fill and stroke (if any) are drawn. If `false`, the mask element's fill and stroke are ignored.

ExportFrameInfo

The following table lists the properties of the `ExportFrameInfo` object, along with their data type and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`delayTime`	integer	For GIF animations, the delay time between frames, in 1/100ths of a second. For example, if you set `delayTime` to `200`, two seconds elapse before the next frame in the animation appears. Default value is `7`.
`frameHidden`	Boolean	If `false` (the default), the frame is exported. If `true`, the frame is hidden and not exported.
`frameName`	string	The name of the frame displayed in the Frames panel. Default is `null`.
`gifDisposalMethod`	string	GIF89a frame disposal method. See the GIF89a specification for details. Acceptable values are `"unspecified"` (the default), `"none"`, `"background"`, and `"previous"`.

ExportOptions

Note: When using this object to set properties, the only required property is `exportFormat`. If other properties are not specified, their default values are used.

The following table lists the properties of the `ExportOptions` object, along with their data types and, where appropriate, acceptable values and notes.

In addition, use the following information to understand the rules for determining scaling in this object.

If `useScale` is `true` (the default), `percentScale` is used to uniformly scale the object on export, and `applyScale` is ignored.

If `useScale` is `false` and `applyScale` is `false` (the default), no scaling is performed on the object on export.

If `useScale` is `false` and `applyScale` is `true`, then `xSize` and `ySize` determine scaling as follows:

- If the value is positive, it specifies the exact size for the axis.
- If the value is zero, it specifies that the axis varies without limit.
- If the value is negative, it specifies that the axis varies, but can be no larger than "`abs(value)`"

If one value is positive and one is negative, the positive value is always used. This gives the following possibilities:

- `xSize < 0, ySize < 0` – use `min(xSize, ySize)` scaling
- `xSize < 0, ySize = 0` – use `xSize` scaling
- `xSize < 0, ySize > 0` – use `ySize` scaling
- `xSize = 0, ySize < 0` – use `ySize` scaling
- `xSize = 0, ySize = 0` – illegal; use `scale` of `1.0`
- `xSize = 0, ySize > 0` – use `ySize` scaling
- `xSize > 0, ySize < 0` – use `xSize` scaling

- `xSize > 0, ySize = 0` – use `xSize` scaling
- `xSize > 0, ySize > 0` – do not use; instead, use `useScale = true` and `percentScale = 0 to 100`

Property	Data type	Notes
`animAutoCrop`	Boolean	Default value is `true`.
`animAutoDifference`	Boolean	Default value is `true`.
`applyScale`	Boolean	Default value is `false`.
`colorMode`	string	Acceptable values are `"indexed"` (the default), `"24 bit"`, and `"32 bit"`.
`crop`	Boolean	Default value is `false`.
`cropBottom`	integer	Default value is `0`.
`cropLeft`	integer	Default value is `0`.
`cropRight`	integer	Default value is `0`.
`cropTop`	integer	Default value is `0`.
`ditherMode`	string	Acceptable values are `"none"` (the default), `"diffusion"`, and `"2 by 2"`.
`ditherPercent`	integer	0 to 100; default value is `100`.
`exportFormat`	string	Acceptable values are `"GIF"`, `"JPEG"`, `"PNG"`, `"custom"`, and `"GIF animation"`. There is no default; this value must be specified.
`frameInfo`	array	Array of `ExportFrameInfo` objects (see "ExportFrameInfo" on page 33); can be `null` (the default).
`interlacedGIF`	Boolean	Default value is `false`.
`jpegQuality`	integer	1 to 100; default value is `80`.
`jpegSmoothness`	integer	0 to 8; default value is `0`.
`jpegSubsampling`	integer	0 to 4; default value is `1`.
`localAdaptive`	Boolean	Default value is `true`.
`lossyGifAmount`	integer	0 to 100; default value is `0`.
`macFileCreator`	string	Default value is `""` (an empty string).
`macFileType`	string	Default value is `""`(an empty string).
`name`	string	Default value is `""` (an empty string).
`numCustomEntries`	integer	0 to 256; default value is `0`.
`numEntriesRequested`	integer	0 to 256; default value is `128`.
`numGridEntries`	integer	0 to 256; default value is `6`.
`optimized`	Boolean	Default value is `true`.
`paletteEntries`	array	Array of color strings (see "Color string" on page 5); default value is `null`.

Property	Data type	Notes
paletteInfo	array	Array of ExportPaletteInfo objects, or null if all entries in the array are default values (see "ExportPaletteInfo" on page 35); default value is null.
paletteMode	string	Acceptable values are "adaptive" (the default), "custom", "grid", "monochrome", "Macintosh", "Windows", "exact", and "Web 216".
paletteTransparencyType	string	Acceptable values are "none" (the default), "index", "index alpha", and "rgba".
percentScale	integer	1 to 100,000; default value is 100.
progressiveJPEG	Boolean	Default value is false.
savedAnimationRepeat	integer	Default value is 0.
sorting	string	Acceptable values are "none" (the default), "luminance", and "popularity".
transparencyIndex	zero-based integer	-1 to 255; pass -1 to use the background color's index; default value is -1.
useScale	Boolean	Default value is true.
webSnapAdaptive	Boolean	Default value is true.
webSnapTolerance	integer	Default value is 14.
xSize	integer	-100,000 to 100,000; default value is 0. See "ExportOptions" on page 33 for details on using xSize and ySize.
ySize	integer	-100,000 to 100,000; default value is 0. See "ExportOptions" on page 33 for details on using xSize and ySize.

ExportPaletteInfo

The following table lists the properties of the ExportPaletteInfo object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
colorLocked	Boolean	true if the color is locked in the panel. Default value is false.
colorModified	Boolean	true if the color was edited. Default value is false.
colorSelected	Boolean	true if the color is selected in the panel (selection is a temporary attribute). Default value is false.
colorTransparent	Boolean	true if the color is exported as transparent. Default value is false.
newColorValue	string	If colorModified is true, specifies the color that will actually be used. Default value is "#000000".

ExportSettings

The following table lists the properties of the `ExportSettings` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`baseName`	string	The name from which all automatically named slice names are derived.
`discardUnspecifiedSlices`	Boolean	If `true`, omits undefined slices from export operations.
`docHtmlEncoding`	string	Determines the encoding standard for the HTML file that Fireworks generates during export. Use "`iso-8859-1`" for ASCII or "`utf-8`" for Unicode.
`docXHTMLFormat`	Boolean	Determines whether Fireworks will output XHTML formatted files (`true`) or HTML formatted files (`false`) when the user exports a file.

`exportFileStyle`	string	Acceptable values are:
		`“HTML and Images”`
		`“Images Only”`
		`“Dreamweaver LBI”`
		`"Director HTML"`
		`"CSS Layers"`
		`"Layers to Files"`
		`"Frames to Files"`
		`"Lotus Domino"`
		`"Macromedia Flash SWF"`
		`"Illustrator"`
		`"Photoshop"`

`fileExtensions`	string	Defines the extension to append to the filename.
`generateDemoHtml`	Boolean	If `true`, generates multiple HTML pages for button export.
`htmlDestination`	string	Acceptable values are "`same`", "`custom`", and "`clipboard`".
`setByUser`	Boolean	If `true`, the user specifies the export settings. If `false`, the first time the file is exported, Fireworks chooses settings based on the data.
`shimGeneration`	string	Acceptable values are "`none`" (no shims), "`transparent`" (one-pixel transparent shims), and "`nested tables`" (no shims, but nested tables).
`sliceAlongGuides`	Boolean	If `true`, use guides for slicing (and `sliceUsingUrls` should be `false`).

Property	Data type	Notes

`sliceAutoNaming1` through `sliceAutoNaming6`	string	Used to generate a name by concatenating six strings. If you need fewer than six strings, fill in the remaining strings with `"none"`.
		Acceptable values are:
		`"none"` – generates nothing.
		`"row_col"` – generates a unique row and column index; `0_0` is first, `0_1` is second, and so on.
		`"ALPHA"` -– generates a unique uppercase letter: `A` is first, `B` is second, and so on.
		`"alpha"` – generates a unique lowercase letter: `a` is first, `b` is second, and so on.
		`"numeric1"` – generates a unique number: `1` is first, `2` is second, and so on.
		`"numeric01"` -– generates a unique two-digit number: `01` is first, `02` is second, and so on.
		`"doc.name"` – name of the file being exported, without a path or extension, such as `"image"`.
		`"slice"` – the string `"slice"`.
		`"underscore"` – the underscore character (_)
		`"period"` – the period character (.)
		`"space"` – the space character ( )
		`"hyphen"` – the hyphen character (-)
		For example, to generate names of `"image_slice01"`, `"image_slice02"`, and so on from a document named `"image"`, set the following properties:
		`sliceAutoNaming1: "doc.name"`
		`sliceAutoNaming2: "underscore"`
		`sliceAutoNaming3: "slice"`
		`sliceAutoNaming4: "numeric01"`
		`sliceAutoNaming5: "none"`
		`sliceAutoNaming6: "none"`

`sliceFrameNaming1` and `sliceFrameNaming2`	string	Used to generate a name by concatenating two strings; the resulting string is concatenated to the name that is specified by `sliceAutoNaming`. If you need fewer than two strings, fill in the remaining string with `"none"`. Acceptable values are: `"none"` – generates nothing. `"frameNumber"` – generates frame number preceded by `f`, for example, `f2`. `"number"` – generates frame number, for example, `2`. `"state"` – generates frame state, for example, `"over"`, `"down"`, or `"overdown"`. `"abbreviation"` – generates abbreviated state, for example, `"o"`, `"d"`, or `"od"`. `"underscore"` – the underscore character (_) `"period"` – the period character (.) `"space"` – the space character ( ) `"hyphen"` – the hyphen character (-)
`sliceUsingUrls`	Boolean	If `true`, use slice objects for slicing (and `sliceAlongGuides` should be `false`).
`templateName`	string	HTML style to be used during export. Acceptable values are `"Dreamweaver"`, `"Generic"`, `"FrontPage"`, `"GoLive"`, or a user-created HTML style.

Fill

The following table lists the properties of the `Fill` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`category`	string	Specifies where this Fill appears in the Fill panel.
`ditherColors`	array	Array of two color strings (see "Color string" on page 5).
`edgeType`	string	Acceptable values are `"hard"` and `"antialiased"`.
`feather`	integer	0 to 1000, which represents the feathering value in pixels (0 means no feathering).
`gradient`	object	`Gradient` object (see "Gradient" on page 39).
`name`	string	The name that appears in the Fill panel.
`pattern`	object	`Pattern` object (see "Pattern" on page 41).
`shape`	string	Acceptable values are `"solid"`, `"linear"`, `"radial"`, `"conical"`, `"satin"`, `"pinch"`, `"folds"`, `"elliptical"`, `"rectangular"`, `"bars"`, `"ripple"`, `"waves"`, `"pattern"`, and `"web dither"`.
`stampingMode`	string	Acceptable values are `"blend"` and `"blend opaque"`.
`textureBlend`	float	0 to 100
`webDitherTransparent`	Boolean	If `true` (and `shape` is `"web dither"`), then the second color in the `ditherColors` array is ignored and transparent is used instead.

Frame

The following table lists the properties of the `Frame` object, along with their data types and, where appropriate, acceptable values and notes. Read-only properties are marked with a bullet (•).

Property	Data type	Notes
`delay`	integer	Hundredths of a second.
`disposal`	string	Acceptable values are `"unspecified"`, `"none"`, `"background"`, and `"previous"`.
`layers` •	array	Array of `FrameNLayerIntersection` objects in the document (see "FrameNLayerIntersection").
`visible`	Boolean	If `false`, this frame is hidden. Default value is `true`.

FrameNLayerIntersection

The following table lists the properties of the `FrameNLayerIntersection` object, along with their data types and, where appropriate, acceptable values and notes. Read-only properties are marked with a bullet (•).

Property	Data type	Notes
`elements` •	array	Array of `Element` objects (see “Element” on page 29).
`locked`	Boolean	If `true`, this FrameNLayerIntersection is locked. Default value is `false`.
`visible`	Boolean	If `false`, this FrameNLayerIntersection is hidden. Default value is `true`.

Gradient

The following table lists the properties of the `Gradient` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`name`	string	The name that appears in the Fill panel.
`nodes`	array	Array of `GradientNode` objects (see “GradientNode” on page 39).
`opacityNodes`	array	Array of `GradientNode` objects (see “GradientNode” on page 39), that identify the opacity ramp that is associated with a gradient.

GradientNode

The following table lists the properties of the `GradientNode` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`color`	string	A color string that specifies the color at this position in the gradient (see “Color string” on page 5).
`isOpacityNode`	Boolean	If true, this node is part of the gradient’s opacity ramp.
`position`	float	0.0 to 1.0

Guides

The following table lists the properties of the `Guides` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`color`	string	A color string that specifies the color that is used for the guides (see “Color string” on page 5).
`hGuides`	array	Array of floating-point numbers that specify horizontal guide locations.

Property	Data type	Notes
locked	Boolean	If true, the user cannot select or move the guides. Default value is false.
vGuides	array	Array of floating-point numbers that specify vertical guide locations.

Layer

The following table lists the properties of the Layer object, along with their data types and, where appropriate, acceptable values and notes. Read-only properties are marked with a bullet (•).

Property	Data type	Notes
disclosure	Boolean	If true, the Layers list displays all the objects in the layer. If false, only the name of the layer appears.
frames •	array	An array of FrameNLayerIntersection objects (see "FrameNLayerIntersection" on page 39).
layerType •	string	Acceptable values are "normal" and "web".
name	string	Might be null (removes any existing name).
sharing	string	Acceptable values are "shared" and "not shared".

PathAttrs

The following table lists the properties of the PathAttrs object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
brush	object	Brush object (see "Brush" on page 21).
brushColor	string	A color string that specifies the color that is used for rendering the Brush object, if any (see "Color string" on page 5).
brushPlacement	string	Acceptable values are "inside", "center", and "outside".
brushTexture	object	Texture object (see "Texture" on page 32).
fill	object	Fill object (see "Fill" on page 38).
fillColor	string	A color string that specifies the color that is used for rendering the Fill object, if any (see "Color string" on page 5).
fillHandle1	point	The three fillHandle properties are used by Gradient and Pattern fills to set the angle and size of the gradient/pattern.
fillHandle2	point	
fillHandle3	point	
fillOnTop	Boolean	If true, the fill is drawn on top of the brush; if false (the default), the fill is drawn beneath the brush.
fillTexture	object	Texture object (see "Texture" on page 32).

Pattern

The following table lists the properties of the `Pattern` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`name`	string	The name that appears in the Fill panel.

RectanglePrimitive

The following table lists the properties and methods of the `RectanglePrimitive` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`roundness`	float	A float value between `0` and `1` that specifies the "roundness" to use for the corners (`0` is no roundness, `1` is 100-percent roundness).
`originalSides`	rectangle	A rectangle that specifies the original sides of the primitive (see "Rectangle" on page 6). Because rectangle primitives remember transformations, the user might see something different from the original sides.
`transform`	matrix	A matrix that indicates all the transformations that were applied to the primitive (see "Matrix" on page 6).
`pathAttributes`	object	A `PathAttrs` object that indicates the path attributes of the primitive (see "PathAttrs" on page 40).

SingleTextRun

The following table lists the properties of the `SingleTextRun` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`changedAttrs`	object	`TextAttrs` object (see "TextAttrs" on page 43).
`characters`	string	The text that is contained in this run.

Style

The following table lists the properties of the `Style` object, along with their data types and, where appropriate, acceptable values and notes. All Style properties are read-only.

Property (read-only)	Data type	Notes
`effectList`	object	`EffectList` object (see "EffectList" on page 29).
`name`	string	The name displayed in the Style panel.
`pathAttributes`	object	`PathAttrs` object (see "PathAttrs" on page 40).
`tdTagText`	string	A string that contains all the attributes of a table cell except `colspan` and `rowspan`. Should be in a format similar to the following: `"bgcolor="ff0000" valign="top"`"

Property (read-only)	Data type	Notes
`textBold`	Boolean	Whether to make the affected text bold; used only if `use_textStyles` is `true`.
`textFont`	string	The font to apply to text; used only if `use_textFont` is `true`.
`textItalic`	Boolean	Whether to make the affected text italic; used only if `use_textStyles` is `true`.
`textSize`	string	String of the form `"#pt"`, where # is a numeric value.
`textUnderline`	Boolean	Whether to underline the affected text; used only if `use_textStyles` is `true`.
`use_brush`	Boolean	If `true`, applies the `brush` property from the `pathAttributes` object when applying the style. If `false`, ignores the `brush` property. Default value is `false`.
`use_brushColor`	Boolean	If `true`, applies the `brushColor` property from the `pathAttributes` object when applying the style. If `false`, ignores the `brushColor` property. Default value is `false`.
`use_effectList`	Boolean	If `true`, applies the `effects` property from the `effectList` object when applying the style. If `false`, ignores the `effects` property. Default value is `false`.
`use_fill`	Boolean	If `true`, applies the `fill` property from the `pathAttributes` object when applying the style. If `false`, ignores the `fill` property. Default value is `false`.
`use_fillColor`	Boolean	If `true`, applies the `fillColor` property from the `pathAttributes` object when applying the style. If `false`, ignores the `fillColor` property. Default value is `false`.
`use_textFont`	Boolean	If `true`, applies the `textFont` property from the `pathAttributes` object when applying the style. If `false`, ignores the `textFont` property. Default value is `false`.
`use_textSize`	Boolean	If `true`, applies the `textSize` property from the `pathAttributes` object when applying the style. If `false`, ignores the `textSize` property. Default value is `false`.
`use_textStyles`	Boolean	If `true`, applies the `textStyles` property from the `pathAttributes` object when applying the style. If `false`, ignores the `textStyles` property. Default value is `false`.

TextAttrs

The following table lists the properties of the `TextAttrs` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`alignment`	string	Acceptable values are `"left"`, `"center"`, `"right"`, `"justify"`, and `"stretch"`.
`baselineShift`	integer	The number of pixels above (positive numbers) or below (negative numbers) the baseline by which the characters are shifted.
`bold`	Boolean	`true` for bold text, `false` for normal text.
`face`	string	The name of the font, such as Arial.
`fillColor`	string	A color string that specifies the color of the text (see "Color string" on page 5).
`horizontalScale`	float	The relative width of the characters. 1.0 – normal width < 1 – thinner than normal > 1 – wider than normal
`italic`	Boolean	`true` for italic text, `false` for normal text.
`kerning`	float	Also known as pair kerning, it is the percentage of an em square by which to separate two characters, in addition to the amount the font specifies. Applies to only one pair or characters. To specify kerning for a range of text, use the `rangeKerning` property. 0 – normal kerning < 0 – move the two characters closer together > 0 – move the two characters farther apart
`leading`	float	The spacing between two lines of text, measured from baseline to baseline. Larger numbers place more space between lines of text. Smaller numbers move the lines closer together. The exact effect of this property number depends on the value of the `leadingMode` property.
`leadingMode`	string	The only acceptable value is `"percentage"`, which specifies that the `leading` property is a percentage of the text's point size. A `leading` property of `1.0` would mean 100 percent or single-spaced, `2.0` would mean 200 percent or double-spaced, and so on.
`rangeKerning`	float	The same as kerning, but applies to a range of text, not only two characters.
`size`	string	String of the form `"#pt"`, where # is a numeric value.
`underline`	Boolean	`true` for underlined text, `false` for normal text.

TextRuns

The following table lists the properties of the `TextRuns` object, along with their data types and, where appropriate, acceptable values and notes.

Property	Data type	Notes
`initialAttrs`	object	`TextAttrs` object (see “TextAttrs” on page 43).
`textRuns`	array	Array of `SingleTextRun` objects on this `TextRuns` object (see “SingleTextRun” on page 41).

HTML export objects

Fireworks provides several object types that support the output of HTML and sliced images from Fireworks. These objects let you write JavaScript scripts that create templates to output the type of HTML that suits your specific requirement (generic HTML, Dreamweaver-compatible HTML, and so on). For each HTML template, use a Slices.htt file that generates the HTML for that particular template. For more information, refer to the Slices.htt and Metafile.htt files that are installed with Fireworks.

Note: For information on how to format nonstandard data types, such as rectangle or point, see “Formatting nonstandard data types” on page 5.

BehaviorInfo

The `BehaviorInfo` object describes a behavior that is assigned to an element. There are seven behaviors: `Status Message, Swap Image, Button Down, Swap Image Restore, Button Highlight, Button Restore`, and `Popup Menu` (new in Fireworks 4). The following table lists the properties of the `BehaviorInfo` object, along with their data types and, where appropriate, acceptable values and notes. All `BehaviorInfo` properties are read-only.

Property (read-only)	Data type	Notes

`action`	integer	Specifies the type of behavior: 1 is Status Message, 2 is Swap Image, 4 is Button Down, 5 is Swap Image Restore, 6 is Button Highlight, 7 is Button Restore, and 9 is Popup Menu. In the standard (default) templates, the following values are defined:
		`var kActionStatusMessage = 1;`
		`var kActionSwapImage = 2;`
		`var kActionButtonDown = 4;`
		`var kActionSwapImageRestore = 5;`
		`var kActionButtonHighlight = 6;`
		`var kActionButtonRestore = 7;`
		`var kActionPopupMenu = 9;`

`downHighlight`	Boolean	For button highlight behaviors, `true` if there is a down highlight image.

`event`	integer	Specifies the type of event: 0 is Mouse Over, 1 is On Click, 2 is Mouse Out, and 3 is On Load. In the standard (default) templates, the following values are defined:
		`var kEventMouseOver = 0;`
		`var kEventOnClick = 1;`
		`var kEventMouseOut = 2;`
		`var kEventOnLoad=3;`

Property (read-only)	Data type	Notes
`hasHref`	Boolean	For swap image behaviors, `true` if the swap image swaps in an external file. The value of `hasHref` is always the opposite of `hasTargetFrame`; you cannot swap from two sources.
`hasStatusText`	Boolean	For status message behaviors, `true` if the status text is not empty.
`hasTargetFrame`	Boolean	For swap image behaviors, `true` if the swap image swaps in another frame in the Fireworks file. The value of `hasTargetFrame` is always the opposite of `hasHref`; you cannot swap from two sources.
`horzOffset`	integer	If `action` is set to 9 (Popup Menu), `horzOffset` specifies the horizontal pixel offset for the menu.
`href`	string	The argument must be which is expressed as a file://URL. For swap image behaviors, the file URL for an external swap image file.
`preload`	Boolean	For swap image behaviors, `true` if the image is to be preloaded.
`restoreOnMouseout`	Boolean	If `true`, the original image for a swap image behavior is restored on mouse out.
`statusText`	string	For status message behaviors, the status message text.
`targetColumnNum`	zero-based integer	For swap image behaviors, the column in the slices table that is swapped.
`targetFrameNum`	zero-based integer	For swap image behaviors, if `hasTargetFrame` is `true`, this frame number is swapped.
`targetRowNum`	zero-based integer	For swap image behaviors, the row in the slices table that is swapped.
`vertOffset`	integer	If `action` is set to 9 (Popup Menu), `vertOffset` specifies the vertical pixel offset for the menu.

BehaviorsList

The `BehaviorsList` object is an array of `BehaviorInfo` objects that describe the behaviors in an image map (see “BehaviorInfo” on page 44). The `BehaviorsList` object does not occur by itself. That is, all occurrences of `BehaviorsList` objects are members of other objects. In the following example, `behaviors` is an object of type `BehaviorsList`, and `curBehavior` is an object of type `BehaviorInfo`.

```
var curBehavior = slices[i][j].behaviors[k];
```

The `BehaviorsList` object has only one property, which is read-only and is shown in the following table.

Property (read-only)	Data type	Notes
`numberOfBehaviors`	integer	The number of `BehaviorInfo` objects in the `BehaviorsList` array (0 or more) (see “BehaviorInfo” on page 44).

exportDoc

The following table lists the properties of the `exportDoc` object, along with their data types and, where appropriate, acceptable values and notes. All `exportDoc` properties are read-only.

Note: This object type does not start with a capital letter.

Property (read-only)	Data type	Notes
`altText`	string	The alternate text description for the Fireworks document.
`backgroundColor`	string	The hex color of the document canvas, without the # character; for example, `"FF0000"` for red background.
`backgroundIsTransparent`	Boolean	`true` if the Fireworks canvas color is transparent or if the export settings specify a transparent GIF format; `false` otherwise.
`backgroundLink`	string	The background URL, which is expressed as a file://URL.
`docID`	integer	A number that is assigned to a document to help identify HTML generated from it. The `docID` does not change when you change the name of a file. However, if you use File > Save As, you can get multiple files with the same `docID`.
`docSaveFolder`	string	`docSaveFolder` contains the path of the directory into which the document was last saved. If the document has not yet been saved, this is an empty string.
`docSaveName`	string	The filename used when the document was saved, without path information, such as `"nav.gif"`.
`emptyCellColor`	string	A color string that specifies the color of empty table cells (see "Color string" on page 5).
`emptyCellContents`	integer	Specifies what to put into empty cells. Acceptable values are 1 (nothing), 2 (spacer image), and 3 (nonbreaking space).
`emptyCellUsesCanvasColor`	Boolean	If `true` (the default), empty cells are set to the `backgroundColor` property. If `false`, they are set to the `emptyCellColor` property.
`filename`	string	URL for the exported image, relative to the HTML output; for example, `"images/Button.gif"`. In the Slices.htt file, it is the base image name plus the base extension. Unless there is only one slice, the Slices.htt file produces filenames such as `"Button_r2_c2.gif"`.
`generateHeader`	Boolean	`true` if an HTML file is generated; `false` if the output goes to the Clipboard.
`hasAltText`	Boolean	`true` if the Fireworks document has an alternate text description.
`hasBackgroundLink`	Boolean	`true` if the Fireworks document has a background URL.
`height`	integer	Height of the image that is being exported, in pixels. In the Slices.htt file, it is the total height of the output images.

Property (read-only)	Data type	Notes
`htmlEncoding`	string	Determines the encoding standard for the HTML file that Fireworks generates during export. Use "`iso-8859-1`" for ASCII or "`utf-8`" for Unicode.
`htmlOutputPath`	string	File that the HTML is being written to, including filename, which is expressed as a file://URL; for example, "`file:///C|/top/nav/navbar.htm`".
`imagename`	string	Name of the image that is being exported, without extension; for example, "`Button`".
`includeHTMLComments`	Boolean	The value of the Include HTML Comments preference, which the export script interprets as appropriate. For example, if this value is `false`, the Dreamweaver export script removes all nonessential comments.
`numFrames`	integer	Number of frames that are being exported from the Fireworks document. This value is not zero-based; the value is 1 or more.
`pathBase`	string	Path of the image that is being exported; for example, "`images/Button`".
`pathSuffix`	string	Filename extension of the image that is being exported, including a period; for example, "`.gif`".
`startColumn`	integer	Used only in the Metafile.htt file for generating HTML for one slice. Specifies the column of the slice.
`startRow`	integer	Used only in the Metafile.htt file for generating HTML for one slice. Specifies the row of the slice.
`style`	string	The HTML style that is used to export the data, such as "`Dreamweaver`", "`Generic`", or "`FrontPage`".
`tableAlignment`	string	A string that contains the alignment of the table. If the table is left-aligned, the string is simply a space (this is used for writing the HTML table). If the table is center-aligned, the string is "`align="center"`". If the table is right-aligned, the string is "`align="right"`".
`width`	integer	Width of the image being exported, in pixels. In the Slices.htt file, it is the total width of the output images.
`xhtmlFormat`	Boolean	Determines whether Fireworks will output XHTML formatted files (`true`) or HTML formatted files (`false`) when the user exports a file.

ImageMap

The following table lists the properties and methods of the ImageMap object, along with their data types and, where appropriate, acceptable values and notes. All ImageMap properties are read-only.

Property (read-only) or Method	Data type	Notes
altText	string	The alternate text description for this slice, if any.
behaviors	object	BehaviorsList object that contains the behaviors for this slice (see “BehaviorsList” on page 45).
hasAltText	Boolean	true if the slice has an alternate text description.
hasHref	Boolean	true if the slice has a URL.
hasTargetText	Boolean	true if the target text is not empty.
href	string	The URL link for this slice. The argument must be which is expressed as a file://URL.
numCoords	integer	Number of coordinates in the area. A circle always has 1 (the center), a rectangle has 2 (top left and bottom right), and a polygon has 1 or more.
radius	integer	Radius of the area, if shape is "circle".
shape	string	Acceptable values are "circle", "poly", and "rect".
targetText	string	Target text for this image, if any.
xCoord(index)	zero-based integer	Returns the *x* coordinate for the specified point, in pixels. For example, the following commands return the coordinates for the first point: `var x = imagemap.xCoord(0);` `var y = imagemap.yCoord(0);` It is possible to have negative values if the image map area is drawn so that it crosses the left or top sides of the image (or sliced image).
yCoord(index)	zero-based integer	Returns the *y* coordinate for the specified point, in pixels. See xCoord().

ImagemapList

The ImagemapList is an array of ImageMap objects that describe the areas in an image map (see “ImageMap” on page 48). To access imageMap objects, use the ImagemapList array, as shown below:

```
var curImagemap = ImagemapList[i];
```

The ImagemapList object has only one property, which is read-only and shown in the following table.

Property (read-only)	Data type	Notes
numberOfURLs	integer	The number of image map areas in the image map list (0 or more).

SliceInfo

The following table lists the properties and methods of the `SliceInfo` object, along with their data types and, where appropriate, acceptable values and notes. All `SliceInfo` properties are read-only.

Property (read-only) or Method	Data type	Notes
`altText`	string	The alternate text description for this slice.
`behaviors`	object	`BehaviorsList` object that contains the behaviors for this slice (see “BehaviorsList” on page 45).
`cellHeight`	integer	Height of this table row in pixels.
`cellWidth`	integer	Width of this table column in pixels.
`downIndex`	zero-based integer	The index for this slice as a button if it is a multiple file button export down.
`getFrameFileName (`*`frameIndex`*`)`	zero-based integer	Returns a string that is the filename for the slice on the specified frame, without directory or extension information. For example, when exporting a file base named `Button`, `Slices[0][0].getFrameFileName(0)` returns `"Button_r1_c1"`. Generally all slices that have images have a frame filename. For frames 1 and higher, only slices that are rollovers or that are targeted by a swap image have names.
`hasAltText`	Boolean	`true` if the slice has an alternate text description.
`hasHref`	Boolean	`true` if the slice has a URL.
`hasHtmlText`	Boolean	`true` if the cell is a text-only slice.
`hasImage`	Boolean	`true` if this cell has an image. For text-only slices, this is `false`.
`hasImagemap`	Boolean	`true` if there are image map hotspots in this image slice.
`hasTargetText`	Boolean	`true` if the target text is not empty.
`height`	integer	Height of the image in pixels, including row spans.
`href`	string	The URL link for this slice. The argument must be which is expressed as a file://URL.
`htmlText`	string	Text for a text-only slice.
`imagemap`	object	`ImagemapList` object containing the image map information for this slice (see “ImagemapList” on page 48).
`imageSuffix`	string	Extension for the image in this cell, including a period (.); for example, `".gif"`.
`isUndefined`	Boolean	`true` if the slice does not have a slice object drawn over it. If you draw two slices that don’t cover your document, Fireworks automatically generates slices to cover the rest of the document. These slices are undefined.
`left`	integer	Left side of the cell in pixels. The left starts at `0`.

Property (read-only) or Method	Data type	Notes
nestedTableSlices	object	Slices object that describes a nested table that occupies the current table cell (see “Slices” on page 50). null if the cell does not contain a nested table.
setFrameFileName (*frameIndex*)	zero-based integer	Sets the filename for the slice on the specified frame, without directory or extension information. You can stop an image from exporting by setting its name to "" (an empty string).
skipCell	Boolean	true if this cell in the table is covered by a previous row span or column span.
targetText	string	Target text for this image, if any.
top	integer	Top of the cell in pixels. The top starts at 0.
width	integer	Width of the image in pixels, including column spans.

Slices

Slices is an object that has some properties and is also a two-dimensional array of SliceInfo objects (see “SliceInfo” on page 49). For example, Slices[0][0] is the SliceInfo for the first cell at row 0, column 0. The first array is rows, the second is columns.

The following example shows a common way to access the table:

```
var curRow;
var curCol;
for (curRow = 0; curRow<slices.numRows; curRow++) {
    for (curCol=0; curCol<slices.numColumns; curCol++) {
        var curSlice = slices[curRow][curCol]; // curSlice is the slice info
  for the cell at this row &
   column.
        // do whatever processing with curSlice.
    }
}
```

The following table lists the properties of the Slices object, along with their data types and, where appropriate, acceptable values and notes. All Slices properties are read-only.

Property (read-only)	Data type	Notes
demoIndex	zero-based integer	Index for each file generated for multiple file button export.
doDemoHTML	Boolean	true for multiple file button rollover export.
doShimEdges	Boolean	true if table shims are set to Transparent Image in Document properties.
doSkipUndefined	Boolean	true if Export Undefined Slices is not checked in Document Properties.
imagesDirPath	string	Relative URL to the images folder. For example , "images/", or "../site_images", or "" (an empty string) if the images and the HTML are in the same directory.
numColumns	integer	Number of columns that are present in the HTML table. Does not include shim column.

Property (read-only)	Data type	Notes
`numRows`	integer	Number of rows that are present in the HTML table. Does not include shim row.
`shimPath`	string	Relative URL to the shim GIF file; for example, `"images/shim.gif"`.

Working with selected objects

When an object is selected, you can return (get) or set the value of that object's properties. In Fireworks, an object is classified as one of the following element types:

- Hotspot
- SliceHotspot (basically, a slice)
- Path
- Group
- Instance
- Text
- RectanglePrimitive
- pathAttributes
- Image

To test to see if a text block is selected, type the following code:

```
firstSelection = fw.selection[0];

if (firstSelection == "[object Text]"){
alert("I am a text block");
}
```

You can use the information in the following sections to return or set property values.

Note: The return value for a property may be `null`.

Working with properties for any selected object

You can return and set the properties in the following list of any type of selected object:

- `top`
- `left`
- `width`
- `height`
- `visible`
- `opacity`
- `blendMode`
- `name`
- `mask`

To return the name of the selected object, type the following code:

```
objectName=fw.selection[0].name;
```

The following properties contain other properties that you can return or set:

elementMask

- `element`
- `linked`
- `enabled`
- `mode`
- `showAttrs`
- `autoExpandImages`

effectList

- `name`
- `effects`

To return the name of the first effect that is applied to the selected object, type the following code:

```
effectName=fw.selection[0].effectList.effects[0].name;
```

Working with specific properties for elements

Some elements have specific properties that can be returned and set in addition to the list of properties in "Working with properties for any selected object" on page 51.

Hotspot

- `shape`
- `urlText`
- `altText`
- `targetText`
- `contour`
- `behaviors` (returns an array of behaviors)
- `color`

To return the `alt` tag that has been applied to the currently selected hotspot, type the following code:

```
altTag = fw.selection[0].altText;
```

SliceHotspot

SliceHotspot is a subclass of Hotspot. A slice has all the Hotspot's properties, plus the following properties:

- `baseName`
- `htmlText`
- `tdTagText`
- `sliceKind` (`"image"` or `"empty"`)
- `exportOptions`
- `sliceID` (read-only)

To return the name of the currently selected slice, type the following code:

```
sliceName = fw.selection[0].baseName;
```

Path

- `pathAttributes`

Note: For the complete list of path attributes properties, see "pathAttributes" on page 54.

- `randSeed`
- `textureOffset`
- `contours`

To return the value of the fill color for the currently selected path, type the following code:

```
fillColor = fw.selection[0].pathAttributes.fillColor
```

Group

- `elements`
- `groupType`

To return the number of objects in a selected group, type the following code:

```
numOfObjectsinGroup = fw.selection[0].elements.length;
```

Instance

- `symbolID`
- `transformMode`
- `instanceType`
- `urlText`
- `altText`
- `targetText`

To return the `instanceType` for the currently selected instance, type the following code:

```
instance = fw.selection[0].instanceType;
```

Text

- `antiAliased`
- `antiAliasMode`
- `autoKern`
- `orientation`
- `pathAttributes`

Note: For the complete list of path attributes properties, see “pathAttributes” on page 54.

- `randSeed`
- `textRuns`
- `textureOffset`
- `transformMode`

To return the `antiAliasMode` setting for the currently selected text block, type the following code:

```
antiAliasedSetting = fw.selection[0].antiAliasMode;
```

RectanglePrimitive

- `Roundness`
- `pathAttributes`

Note: For the complete list of path attributes properties, see “pathAttributes” on page 54.

- `originalSides`
- `transform`

To return the roundness setting for the currently selected rectangle, type the following code:

```
roundness = fw.selection[0].roundness;
```

pathAttributes

Several objects have the `pathAttributes` property. The following list is the valid set of `pathAttributes` subproperties that can be returned or set:

- `brushColor`
- `fillColor`
- `brush`
- `fill`
- `brushTexture`
- `fillTexture`
- `fillHandle1`
- `fillHandle2`

- `fillHandle3`
- `brushPlacement`
- `fillOnTop`

To return the name of brush on the current path, type the following code:

```
brush = fw.selection[0].pathAttributes.brush.name;
```

CHAPTER 3
Fireworks JavaScript API

To make it possible to create useful Fireworks extensions and customized Fireworks menus, Fireworks supports the JavaScript functions that are listed in this chapter. Almost any task that the user can accomplish in Fireworks with the menus, tools, or floating panels can now be done using JavaScript.

Using Fireworks API functions

Three categories of API functions are described in this chapter: Document functions, History panel functions, and Fireworks functions. The following rules apply to all functions.

Zero-based indexes

Some functions take an *`index`* argument which is a zero-based-one-dimensional array. That means a value of `0` represents the first item in the array, `1` represents the second item, and so on. For example, the following command deletes the second layer of the active Fireworks document:

```
fw.getDocumentDOM().deleteLayer2;
```

Functions that take a *`frameIndex`* argument can be passed `-1` to indicate the current frame. Similarly, functions that take a *`layerIndex`* argument may be passed `-1` to indicate the current layer.

Passing null values

In general, passing a `null` value to a function causes an exception to be thrown. A few functions do allow `null` as an argument; such cases are noted in the function descriptions.

Operating on a selection

Many API functions in this chapter refer to a "selection" or to "selected items." These terms refer to Fireworks elements, such as text boxes or images, that are currently selected. In most cases, the functions work even if only one item is selected. If a function requires more than one selected item, this is noted in the description of the function.

Palette or panel

Several API functions reference the History panel (see "History panel functions" on page 197). Throughout the Fireworks documentation and online help, the term "palette" is reserved for discussions of a color palette, and the term "panel" is used to refer to the floating windows that are available within Fireworks. Therefore, when the function name contains "palette," the descriptions refer to a "panel."

Document functions

As discussed in "Accessing a Fireworks document" on page 7, you get and set document properties by calling functions as methods of the document's Document Object Model (DOM). Methods that operate on a document's DOM are listed in this section as `dom.functionName()`. However, you cannot simply type `dom.functionName()`. In place of `dom`, you must type `fw.getDocumentDOM()` or `fw.documents[documentIndex]`. For example:

- How a function looks in this manual: `dom.addNewHotspot()`
- How you must type it:

  ```
  fw.getDocumentDOM().addNewHotspot(); // operates on active document
  ```

  or

  ```
  fw.documents[documentIndex].addNewHotspot(); // operates on specified
    document
  ```

dom.addBehavior()

Availability

Fireworks 3

Description

Adds a specified behavior to the selected hotspots and slices.

Arguments

`action, event, eventIndex`

- *`action`* is a string that specifies the behavior to be added, such as `"MM_swapImageRestore()"`. For a list of all the behaviors that can be added, see "Using the addBehavior() function" on page 201.
- *`event`* specifies the event that triggers the behavior. Acceptable values are `"onMouseOver"`, `"onMouseOut"`, `"onLoad"`, and `"onClick"`.
- *`eventIndex`* is a zero-based integer that specifies the location where the behavior should be added. To specify the end location, pass `-1` here.

Returns

Nothing.

Example

The following command adds a simple rollover behavior at the end of the selected slice or hotspot.

```
fw.getDocumentDOM().addBehavior("MM_simpleRollover()", "onMouseOver", -1);
```

Related functions

`dom.removeBehavior()`

dom.addElementMask()

Availability

Fireworks 4

Description

Adds a new empty mask to the selected element. If the selection already has an element mask, it is replaced with the new one. Only one element can be selected when calling this function. If selecting more than one element (or none) at the time this function is called, Fireworks throws an exception.

Arguments

mode, {bEnterMaskEditMode}

- Acceptable values for *mode* are `"reveal all"`, `"hide all"`, `"reveal selection"`, and `"hide selection"`. If the user is not in bitmap mode, or if there is no pixel selection, `"reveal selection"` and `"hide selection"` operate the same as `"reveal all"` and `"hide all"`, respectively.
- If *{bEnterMaskEditMode}* (optional) is `true`, Fireworks enters mask-edit mode on the newly added mask; if omitted, it defaults to `false`.

Returns

Nothing.

dom.addFrames()

Availability

Fireworks 3, enhanced in 4

Description

Adds one or more frames to the document.

Arguments

howMany, where, {bAdvanceActiveFrame}

- *howMany* is an integer that specifies how many frames to add.
- *where* specifies where to add the frames. Acceptable values for *where* are `"beginning"`, `"before current"`, `"after current"`, and `"end"`.
- (*bAdvanceActiveFrame*), which was added in Fireworks 4, specifies whether to change the active frame. If it is omitted or `true`, this function sets the active frame to the first frame added. If `false`, the active frame does not change. For example, if the user is adding frames at the end of a document that has two frames and *bAdvanceActiveFrame* is omitted or `true`, then the third frame becomes the active frame.

Returns

Nothing.

Example

The following command adds one frame after the current frame but does not change the active frame.

```
fw.getDocumentDOM().addFrames(1, "after current", false);
```

dom.addGuide()

Availability

Fireworks 3

Description

Adds a guide to the document. If a guide already exists at the specified position, this function has no effect.

Arguments

`position, guidekind`

- `position` is a float value that specifies the *x* or *y* coordinate at which to add the guide.
- Acceptable values for `guidekind` are `"horizontal"` and `"vertical"`. If *guidekind* is `"horizontal"`, it is assumed that `position` is a *y* coordinate; if `"vertical"`, it is an *x* coordinate.

Returns

Nothing.

Example

The following command adds a vertical guide at the *x* coordinate of 217.

```
fw.getDocumentDOM().addGuide(217, "vertical");
```

dom.addNewHotspot()

Availability

Fireworks 3

Description

Adds a new hotspot that fits into the specified bounding rectangle.

Arguments

`hotspot-kind, hotspot-shape, boundingRectangle`

- `hotspot-kind` can be `"hotspot"` or `"slice"`.
- `hotspot-shape` can be `"rectangle"` or `"oval"`.
- `boundingRectangle` is a rectangle that specifies the bounds within which the hotspot is placed (see “Rectangle” on page 6).

Returns

Nothing.

Example

The following command adds a new rectangle slice with the specified coordinates.

```
fw.getDocumentDOM().addNewHotspot("slice","rectangle",{left:0, top:0,
  right:50, bottom:100});
```

dom.addNewImage()

Availability

Fireworks 3

Description

Adds a new empty (transparent) image to the document.

Arguments

boundingRectangle, *bEnterPaintMode*

- *boundingRectangle* is a rectangle that specifies the bounds of the image to be added (see “Rectangle” on page 6). You cannot create an image that is larger than the document; therefore, if you pass in a rectangle with bounds larger than the document size, you can create an image that is constrained to the document size.
- If *bEnterPaintMode* is `true`, the application immediately enters bitmap mode for the new image.

Returns

Nothing.

Example

The following command adds an empty image that is 500 by 500 pixels in size, and then enters bitmap mode.

```
fw.getDocumentDOM().addNewImage({left:0, top:0, right:500, bottom:500}, true);
```

dom.addNewImageViaCopy()

Availability

Fireworks MX

Description

Adds a new image to the document containing the contents of the current paint-mode selection. The new image is placed directly above the active bitmap. You must have a current pixel selection for this to succeed. The new bitmap appears with Fireworks in paint mode.

Arguments

None.

Returns

Nothing.

dom.addNewImageViaCut()

Availability

Fireworks MX

Description

Adds a new image to the document that contains the contents of the current paint mode selection. The new image is placed directly above the active bitmap. You must have a current pixel selection for this to succeed. The selection is cut from the previously active bitmap. The new bitmap appears with Fireworks in paint mode.

Arguments

None.

Returns

Nothing.

dom.addNewLayer()

Availability

Fireworks 3

Description

Adds a new layer to the document and makes it the current layer.

Arguments

`name, bShared`

- *`name`* is a string that specifies the name for the new layer. If *`name`* is `null`, a new layer name is generated.
- *`bShared`* is a Boolean value that specifies whether the new layer is shared.

Returns

A string value that contains the name of the new layer.

Example

The following command adds a new unshared layer with a default name that is generated by Fireworks.

```
fw.getDocumentDOM().addNewLayer(null, false);
```

dom.addNewLine()

Availability

Fireworks 3

Description

Adds a new path between two points. The new path uses the document's current default path attributes and is added to the current frame and layer.

Arguments

`startPoint, endPoint`

`startPoint` and *`endPoint`* are points that specify the *x,y* coordinates between which the path is added (see "Point" on page 6).

Returns

Nothing.

Example

The following command adds a new line between the specified coordinates.

```
fw.getDocumentDOM().addNewLine({x:64.5, y:279.5}, {x:393.5, y:421.5});
```

dom.addNewOval()

Availability

Fireworks 3

Description

Adds a new oval fitting into the specified bounding rectangle. The oval uses the document's current default path attributes and is added on the current frame and layer.

Arguments

`boundingRectangle`

`boundingRectangle` is a rectangle that specifies the bounds of the oval to be added (see "Rectangle" on page 6).

Returns

Nothing.

Example

The following command adds a new oval within the specified coordinates.

```
fw.getDocumentDOM().addNewOval({left:72, top:79, right:236, bottom:228});
```

dom.addNewRectangle()

Availability

Fireworks 3

Description

Adds a new rectangle or rounded rectangle fitting into the specified bounds. The rectangle uses the document's current default path attributes and is added on the current frame and layer.

Arguments

`boundingRectangle, roundness`

- `boundingRectangle` is a rectangle that specifies the bounds within which the new rectangle is added (see "Rectangle" on page 6).
- `roundness` is a float value between 0 and 1 that specifies the "roundness" to use for the corners (0 is no roundness, 1 is 100 percent roundness).

Returns

Nothing.

Example

The following command adds a new rectangle with no round corners within the specified coordinates.

```
fw.getDocumentDOM().addNewRectangle({left:0, top:0, right:100, bottom:100},
  0);
```

Related functions

`dom.addNewRectanglePrimitive()`

dom.addNewRectanglePrimitive()

Availability

Fireworks 4

Description

Adds a new rectangle primitive that fits into the specified bounds. The rectangle primitive uses the document's current default path attributes, is added on the current frame and layer, and has several editable properties, such as corner roundness and transformation. The difference between a rectangle and a rectangle primitive is that a rectangle is a path that is shaped like a rectangle, and a rectangle primitive remembers its "rectangleness"; that is, if you drag a corner, it remains a rectangle, rather than deforming into a quadrilateral.

Arguments

`boundingRectangle`, *`roundness`*

- *`boundingRectangle`* is a rectangle that specifies the bounds within which the new rectangle primitive is added (see "Rectangle" on page 6).
- *`roundness`* is a float value between 0 and 1 that specifies the "roundness" to use for the corners (0 is no roundness, and 1 is 100 percent roundness).

Returns

Nothing.

Example

The following command adds a new rectangle primitive with no round corners within the specified coordinates.

```
fw.getDocumentDOM().addNewRectanglePrimitive({left:0, top:0, right:100,
  bottom:100}, 0);
```

Related functions

`dom.addNewRectangle()`, `fw.ungroupPrimitives()`

dom.addNewSinglePointPath()

Availability

Fireworks 3

Description

Adds a new path that consists of a single Bézier point. The path uses the default fill, stroke, and so on, and is added on the current frame and layer. The point is selected after it is added.

Arguments

`controlPointFirst`, *`mainPoint`*, *`controlPointLast`*, *`bCopyAttrs`*

- *`controlPointFirst`*, *`mainPoint`*, and *`controlPointLast`* are points that specify the *x,y* coordinates of the preceding control point, the main point, and the following control point of the Bézier path (see "Point" on page 6).
- If *`bCopyAttrs`* is `false`, the path's stroke and fill are copied directly from the document's current stroke and fill settings. If it is `true`, the path's fill is set to None, and the brush is set to something other than None.

Returns

Nothing.

Example

The following command adds a new path that consists of a single Bézier point at the specified coordinates and copies the path's stroke and fill from the document's current stroke and fill settings.

```
fw.getDocumentDOM().addNewSinglePointPath({x:150, y:63}, {x:150, y:63},
  {x:150, y:63}, false);
```

dom.addNewStar()

Availability

Fireworks 3

Description

Adds a new star- or polygon-shaped path.

Arguments

`numSides`, *`spikiness`*, *`bIsStar`*, *`centerPoint`*, *`outsidePoint`*

- *`numSides`* is an integer that specifies the number of sides of the new path.
- *`spikiness`* is a float value that controls the regularity of the star or polygon. Pass `-1` to have Fireworks calculate a good value, or pass a value between 0 and 1 for manual control.
- If *`bIsStar`* is `true`, a star with the specified number of points is created. If it is `false`, a regular polygon with the specified number of sides is created.
- *`centerPoint`* specifies the center point of the star or polygon (see "Point" on page 6).
- *`outsidePoint`* specifies a point on the radius of the star or polygon.

Returns

Nothing.

Example

The following command adds a five-sided star.

```
fw.getDocumentDOM().addNewStar(5, -1, true, {x:186, y:72}, {x:265, y:89});
```

dom.addNewSymbol()

Availability

Fireworks 3

Description

Adds a new symbol to the library and opens the symbol document for editing. Optionally adds an instance of the symbol to the document.

Arguments

`type`, *`name`*, *`bAddToDoc`*

- *`type`* can be `"graphic"`, `"button"`, or `"animation"`.
- *`name`* is a string that specifies the name of the symbol.
- If *`bAddToDoc`* is `true`, an instance of the symbol is inserted into the center of the document. If `false`, the symbol is created in the document's library, but no instance of the symbol is inserted into the document.

Returns

Nothing.

Example

The following command adds a new graphic symbol called `text` to the library and places an instance of it in the document.

```
fw.getDocumentDOM().addNewSymbol("graphic", "text", true);
```

dom.addNewText()

Availability

Fireworks 3

Description

Adds a new empty text block within the specified bounding rectangle. (To place text in the box, use `dom.setTextRuns()`.)

Arguments

`boundingRectangle, bInitFromPrefs`

- *`boundingRectangle`* is a rectangle that specifies the bounds within which to place the new text box (see “Rectangle” on page 6).
- If *`bInitFromPrefs`* is `false`, the default values for all style properties are used. If it is `true`, the most recent values set by the user are used.

Returns

Nothing.

Example

The following command adds a text box with the most recently used style properties.

```
fw.getDocumentDOM().addNewText({left:43, top:220, right:102, bottom:232},
  true);
```

dom.addSwapImageBehaviorFromPoint()

Availability

Fireworks 3

Description

If a single hotspot or slice is selected, this function adds to it a swap image behavior from the hotspot or slice located at *`where`* in the document.

Arguments

`where`

`where` is a point that specifies the *x,y* coordinates of the hotspot or slice that contains the swap image behavior to be added (see “Point” on page 6).

Returns

`true` if the swap image behavior was added; `false` if no suitable hotspot was at the specified location.

dom.adjustExportToSize()

Availability

Fireworks 3

Description

Adjusts the export settings as specified.

Arguments

`sizeInBytes, bOkToIncreaseSize`

- *`sizeInBytes`* is an integer that specifies the size to be used for exporting. It is used as described in the following list:

  If a document has no slices, *`sizeInBytes`* adjusts the export settings for the current frame so that the image is less than or equal to *`sizeInBytes`*.

  If a document has slices, *`sizeInBytes`* adjusts the size of all exported images so that the sum of the sizes is greater than or equal to *`sizeInBytes`*.

- *`bOkToIncreaseSize`* specifies whether the export file size can be increased.

  If *`bOkToIncreaseSize`* is `true`, and the current size is less than ***sizeInBytes***, the argument increases the quality of the export settings as much as possible, making the export size larger if necessary.

  If *`bOkToIncreaseSize`* is `false`, the argument increases the quality of the export settings as much as possible without increasing the export size.

dom.adjustFontSize()

Availability

Fireworks MX

Description

Increases (positive values) or decreases (negative values) the font size of selected text elements. If a text element has multiple font sizes, each size is adjusted independently.

Arguments

`amount`

`amount`, which is specified in points, changes the font size. Positive values (such as `"2pt"`) increase the size, while negative values (such as `"-1pt"`) decrease the size.

Returns

Nothing.

dom.align()

Availability

Fireworks 3

Description

Aligns the selection.

Arguments

`alignmode`

Acceptable values for *`alignmode`* are `"left"`, `"right"`, `"top"`, `"bottom"`, `"center vertical"`, and `"center horizontal"`.

Returns

Nothing.

dom.appendPointToHotspot()

Availability

Fireworks 3

Description

Appends a point to the selected unclosed polygon hotspot. If an unclosed polygon hotspot is not selected, a new polygon hotspot is created with the single point that passed in.

Arguments

`pt, tolerance`

- *`pt`* is a point that specifies the *x,y* coordinates of the point to be added (see "Point" on page 6).
- *`tolerance`* is a float value > = 0 that specifies the tolerance between the new point and the starting point of the polyline path. If the new point is within *`tolerance`* of the starting point, the polyline path is closed.

Returns

Nothing.

dom.appendPointToPath()

Availability

Fireworks 3

Description

Appends a Bézier point to the selected path.

Arguments

`contourIndex, ptToInsertBefore, controlPointFirst, mainPoint, controlPointLast`

- *`contourIndex`* is a zero-based integer that specifies the contour to which the Bézier point is appended. For paths with multiple contours, the contours are in an arbitrary order.
- *`ptToInsertBefore`* is a zero-based integer that specifies where on the path the new point should be placed. The new point is appended in front of the point that this integer represents. To add a point to the beginning of the path, pass `0`; to add a point to the end of the path, pass a large number.
- *`controlPointFirst`*, *`mainPoint`*, and *`controlPointLast`* are points that specify the *x,y* coordinates of the preceding control point, the main point, and the following control point of the new point (see "Point" on page 6).

Returns

Nothing.

Related functions

`dom.insertPointInPath()`

dom.appendPointToSlice()

Availability

Fireworks 3

Description

Appends a point to the selected unclosed polygon slice. If an unclosed polygon slice is not selected, then a new polygon slice is created with the single point that passed in.

Arguments

`pt, tolerance`

- *`pt`* is a point that specifies the *x,y* coordinates of the point to be added (see “Point” on page 6).
- *`tolerance`* is a float value > = 0 that specifies the tolerance between the new point and the starting point of the polyline path. If the new point is within *`tolerance`* of the starting point, the polyline path is closed.

Returns

Nothing.

dom.applyCharacterMarkup()

Availability

Fireworks 3, enhanced in 4

Description

Applies the specified character markup to the selected text.

Arguments

`tag`

Acceptable values for *`tag`* are `"b"`, `"i"`, and `"u"`, for bold, italic, and underline: and `"fwplain"`, which was added in Fireworks 4, for text with no character markup.

Returns

Nothing.

dom.applyCurrentFill()

Availability

Fireworks 3

Description

Applies the document’s current fill to the selection.

Arguments

`bNoNullFills`

If *`bNoNullFills`* is `true` and the current fill is None, then a default fill is applied instead of no fill.

Returns

Nothing.

Example

The following command applies the current fill to the selection.

```
fw.getDocumentDOM().applyCurrentFill(true);
```

dom.applyEffects()

Availability

Fireworks 3

Description

Applies the specified effects to the selection.

Arguments

`effectList`

- *`effectList`* is an `EffectList` object (see “EffectList” on page 29).
- If *`effectList`* is `null`, this function removes all effects from the selection.

Returns

Nothing.

Example

The following command applies a drop shadow with an angle of 315, a blur of 4, a color of black, and a distance of 7 (see “Drop Shadow” on page 26).

```
fw.getDocumentDOM().applyEffects({category:"Untitled", effects:[ {
   EffectIsVisible:true, EffectMoaID:"{a7944db8-6ce2-11d1-8c76000502701850}",
   ShadowAngle:315, ShadowBlur:4, ShadowColor:"#000000a6", ShadowDistance:7,
   ShadowType:0, category:"Shadow and Glow", name:"Drop Shadow" } ],
   name:"Untitled" });
```

dom.applyFontMarkup()

Availability

Fireworks 3

Description

Applies the specified font markup attribute to the selected text.

Arguments

`fontAttribute, value`

- Acceptable values for *`fontAttribute`* are `"size"` and `"face"`.
- If *`fontAttribute`* is `"size"`, *`value`* must be of the form `"XXXpt"` to specify a point size; a simple numeric value is not allowed.

Returns

Nothing.

dom.applyStyle()

Availability

Fireworks 3

Description

Applies the specified style to the selection.

Arguments

`styleName, styleIndex`

- *`styleName`* is a string that specifies the style name to be applied.
- *`styleIndex`* is usually zero. However, if there are multiple styles with the same name, *`styleIndex`* is used to resolve the ambiguity (0 references the first style with that name, 1 references the second, and so on).

Returns

Nothing.

Example

The following command applies the first style that Fireworks encounters named "Style 7", which, in this case, is a default style.

```
fw.getDocumentDOM().applyStyle("Style 7", 0);
```

dom.arrange()

Availability

Fireworks 3

Description

Arranges the selection.

Arguments

`arrangemode`

Acceptable values for *`arrangemode`* are `"back"`, `"backward"`, `"forward"`, and `"front"`.

Returns

Nothing.

Example

The following command brings the selected items to the front.

```
fw.getDocumentDOM().arrange("front");
```

dom.attachTextToPath()

Availability

Fireworks 3

Description

Attaches the selected text to the selected path. If no text and path are selected, no action occurs.

Arguments

None.

Returns

Nothing.

Example

When two items are selected (one a text block and the other a shape), the following command attaches the text block to the shape's path.

```
fw.getDocumentDOM().attachTextToPath();
```

dom.changeGuide()

Availability

Fireworks 3

Description

Moves a guide's position to a new location.

Arguments

`currentPosition, newPosition, guidekind`

- *`currentPosition`* is a float value that specifies the current position of the guide.
- *`newPosition`* is a float value that specifies the new position of the guide.
- Acceptable values for *`guidekind`* are `"horizontal"` and `"vertical"`. If *`guidekind`* is `"horizontal"`, it is assumed that the specified positions are *y* coordinates; if *`guidekind`* is "vertical", it is assumed that the specified positions are *x* coordinates.

Returns

Nothing.

Example

The following command moves a vertical guide from position 135 to position 275.

```
fw.getDocumentDOM().changeGuide(135, 275, "vertical");
```

dom.changeSliceGuide()

Availability

Fireworks MX

Description

Moves a slice guide's position to a new location, which resizes any rectangular slices that abut the guide. A parameter controls whether slice guides that exist between the old position and the new one are also moved.

If a slice is resized so that it has zero width or height, the slice is deleted.

This function does not change slices that are not rectangular.

Arguments

`currentPosition`, *`newPosition`*, *`guidekind`*, *`isMagneticDrag`*

- *`currentPosition`* is a float value that specifies the current position of the slice guide to be moved.
- *`newPosition`* is a float value that specifies the new position of the slice guide.

- `guidekind` accepts values of "`horizontal`" or "`vertical`". If `guidekind` is "`horizontal`", Fireworks assumes that the specified positions are *y* coordinates; if "`vertical`", the specified positions are *x* coordinates.
- `isMagneticDrag` is a Boolean value that determines whether to move other slice guides between the old and new positions. If `isMagneticDrag` is `true`, Fireworks also moves slice guides between the old guide position and the new position. This action resizes and possibly deletes rectangular slices that do not abut the slice guide at `currentPosition`.

Returns

Nothing.

Example

The following command moves a vertical slice guide from position 135 to position 275, and moves all vertical slice guides between 135 and 275 to 275.

```
fw.getDocumentDOM().changeGuide(135, 275, "vertical", true);
```

dom.clearJPEGMask()

Availability

Fireworks 4

Description

Clears the "Selective JPEG mask" for the document.

Arguments

None.

Returns

Nothing.

dom.clipCopy()

Availability

Fireworks 3

Description

Copies the selection to the Clipboard.

Arguments

None.

Returns

Nothing.

Example

The following command copies the selected items to the Clipboard.

```
fw.getDocumentDOM().clipCopy();
```

dom.clipCopyAsPaths()

Availability

Fireworks MX

Description

Copies the selection to the Clipboard in Adobe Illustrator format.

Arguments

None.

Returns

Nothing

Example

The following command copies the selected items to the Clipboard in Adobe Illustrator format.

```
fw.getDocumentDOM().clipCopyAsPaths();
```

dom.clipCopyFormats()

Availability

Fireworks MX

Description

Copies the selection to the Clipboard using the specified format.

Arguments

`format`

`format` defines the graphic format for the selection. For example, `"AICB"` is the Adobe Illustrator format.

Returns

Nothing

dom.clipCut()

Availability

Fireworks 3

Description

Cuts the selection to the Clipboard.

Arguments

None.

Returns

Nothing.

Example

The following command cuts the selected items and places them on the Clipboard.

```
fw.getDocumentDOM().clipCut();
```

dom.clipPaste()

Availability

Fireworks 3, enhanced in 4

Description

Pastes the Clipboard contents into the document.

Arguments

`{`*`whatIfResolutionDifferent, whatIfPastingIntoElementMask`*`}`

- *`whatIfResolutionDifferent`* is an optional string that specifies how resampling should be done if the resolution of the Clipboard contents doesn't match the resolution of the document. Acceptable values for *`whatIfResolutionDifferent`* are `"resample"`, `"do not resample"`, and `"ask user"` (displays a dialog box to let the user decide). If *`whatIfResolutionDifferent`* is omitted or `null`, `"ask user"` is assumed.
- *`whatIfPastingIntoElementMask`*, which was added in Fireworks 4, applies only if the user is editing an element mask, and that element mask is an empty image mask. In this case, the pasted elements will replace the existing mask (because it is essentially a mask that doesn't mask anything). If the image mask isn't empty, the pasted elements are added to the existing mask, rather than replacing it.
- Acceptable values for *`whatIfPastingIntoElementMask`* are `"image", "vector",` and `"ask user"`. If *`whatIfPastingIntoElementMask`* is omitted or `null`, `"ask user"` is assumed.

Returns

Nothing.

Example

The following command pastes the Clipboard contents into the document. If there is a need for resampling, Fireworks asks the user to decide how to resample.

```
fw.getDocumentDOM().clipPaste();
```

dom.clipPasteAsMask()

Availability

Fireworks 4

Description

Pastes the Clipboard contents into the document as an element mask. Only one element can be selected when calling this function. If selecting more than one element (or none) when this function is called, Fireworks throws an exception. An exception is also thrown if there is nothing on the Clipboard.

Arguments

whatIfResolutionDifferent, *masktype*, *maskReplaceOptions*

- *whatIfResolutionDifferent* is a string that specifies how resampling should be done if the resolution of the Clipboard contents doesn't match the resolution of the document. Acceptable values for *whatIfResolutionDifferent* are `"resample"`, `"do not resample"`, and `"ask user"` (displays a dialog box to let the user decide). If *whatIfResolutionDifferent* is omitted or `null`, `"ask user"` is assumed.
- *masktype* specifies how to paste the mask. Acceptable values are `"image"` (always paste as an image mask), `"vector"` (always paste as a vector mask), and `"ask"` (displays a dialog box to let the user decide). If the Clipboard contains a single image, it is pasted as an image mask, even if you pass `"vector"`.
- Acceptable values for *maskReplaceOptions* are `"replace"` (if an element mask already exists, replace it with the pasted one), `"add"` (if an element mask already exists, add the pasted mask to it), and `"ask"` (displays a dialog box to let the user decide).

Returns

Nothing.

dom.clipPasteAttributes()

Availability

Fireworks 3

Description

Pastes the attributes from the Clipboard onto the selection.

Arguments

None.

Returns

Nothing.

Example

The following command applies the attributes that were copied to the Clipboard onto the selected items.

```
fw.getDocumentDOM().clipPasteAttributes();
```

dom.clipPasteFromChannelToChannel()

Availability

Fireworks MX

Description

Pastes the specified color channel on the Clipboard into each of the RGB channels of a new image or into the specified channel of the selected image, if any.

Arguments

`fromChannel, toChannel`

- If the current selection is not a single bitmap, a new opaque bitmap is created and the *`fromChannel`* is pasted in to all three color channels of the new bitmap, resulting in a grayscale image. This first argument is ignored if the current selection is not a single bitmap.
- If the currently selected element is a bitmap, the *`toChannel`* argument is used to specify where to paste the color data.

Returns

Nothing.

Example

The following command copies the red data from the Clipboard into the red channel:

```
fw.getDocumentDOM().clipPasteFromChannelToChannel("red", "red");
```

The following command copies the green data from the clipboard into the alpha channel:

```
fw.getDocumentDOM().clipPasteFromChannelToChannel("green", "alpha");
```

dom.clipPasteInside()

Availability

Fireworks 3, deprecated in 4 in favor of `dom.clipPasteAsMask()` (see "dom.clipPasteAsMask()" on page 75)

Description

Pastes the Clipboard contents into the selection, and makes the selected element into the element mask for the pasted element(s). If the selected element already has a mask, this function groups the pasted elements with the selected element and applies the existing element mask to the group.

Arguments

`{`*`whatIfResolutionDifferent`*`}`

- *`whatIfResolutionDifferent`* is an optional string that specifies how resampling should be done if the resolution of the Clipboard contents doesn't match the resolution of the document. Acceptable values for *`whatIfResolutionDifferent`* are `"resample"`, `"do not resample"`, and `"ask user"` (displays a dialog box to let the user decide).
- If *`whatIfResolutionDifferent`* is omitted or `null`, `"ask user"` is assumed.

Returns

Nothing.

Example

The following command pastes the Clipboard contents inside the selected items. If the resolution of the Clipboard doesn't match the resolution of the document, Fireworks resamples the Clipboard contents to match the document.

```
fw.getDocumentDOM().clipPasteInside("resample");
```

dom.cloneSelection()

Availability

Fireworks 3

Description

Makes exact duplicates of the selection, placing the duplicated items directly on top of the original items.

Arguments

None.

Returns

Nothing.

Example

The following command copies the selected items on top of the original items.

```
fw.getDocumentDOM().cloneSelection();
```

Related functions

`dom.duplicateSelection()`

dom.close()

Availability

Fireworks 3

Description

Closes the document.

Arguments

`bPromptToSaveChanges`

If *`bPromptToSaveChanges`* is `true`, and the document was changed since the last time it was saved, the user is prompted to save any changes to the document. If *`bPromptToSaveChanges`* is `false`, the user is not prompted, and changes to the document are discarded.

dom.convertAnimSymbolToGraphicSymbol()

Availability

Fireworks 4

Description

If a single animation symbol is selected, this function converts it from an animation symbol to a graphics symbol.

Arguments

None.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`, `dom.convertToSymbol()`

dom.convertToAnimSymbol()

Availability

Fireworks 4

Description

Converts the selected item(s) to a new animation symbol.

Arguments

name, *numFrames*, *offsetDistPt*, *rotationAmount*, *scaleAmount*, *startOpacity*, *endOpacity*

- *name* is a string that specifies a name for the new animation symbol.
- *numFrames* is an integer that specifies the number of frames through which the symbol animates.
- *offsetDistPt* is a point that specifies the distance the animation will move in pixels (see "Point" on page 6). For example, passing ({*x*:`100`, *y*:`25`}) animates the symbol to the right 100 pixels and down 25 pixels.
- *rotationAmount* is a float value that specifies the degrees of rotation to be applied to the animation symbol. For example, passing `720` specifies an animation that does two complete clockwise rotations. To rotate the animation counter-clockwise, pass a negative number.
- *scaleAmount* is a positive float value that specifies the amount of scaling to be applied to the animation symbol. For example, passing `50` scales the symbol to 50 percent of its current size, and passing `200` scales it to twice its current size. To specify no scaling, pass `100`.
- *startOpacity* and *endOpacity* are float values between 0 and 100 that specify the starting and ending opacity for the animation symbol.

Returns

Nothing.

Related functions

`dom.convertAnimSymbolToGraphicSymbol()`, `dom.convertToSymbol()`, `dom.setAnimInstanceNumFrames()`

dom.convertToPaths()

Availability

Fireworks 3

Description

Converts the selected text items into editable paths.

Arguments

None.

Returns

Nothing.

Example

The following command converts the selected text items into editable paths.

```
fw.getDocumentDOM().convertToPaths();
```

dom.convertToSymbol()

Availability

Fireworks 3

Description

Converts the selected item(s) to a new symbol.

Arguments

`type, name`

- Acceptable values for *`type`* are `"graphic"`, `"button"`, and `"animation"`.
- *`name`* specifies a name for the new symbol.

Returns

Nothing.

Example

The following command creates a graphic symbol from the selected item and names it "star".

```
fw.getDocumentDOM().convertToSymbol("graphic", "star");
```

Related functions

`dom.convertToAnimSymbol()`, `dom.convertAnimSymbolToGraphicSymbol()`

dom.copyHtmlWizard()

Availability

Fireworks MX

Description

Launches the Copy HTML Wizard dialog box.

Arguments

None.

Returns

Nothing.

Example

The following command launches the Copy HTML Wizard dialog box:

```
fw.getDocumentDOM().copyHtmlWizard();
```

dom.copyToHotspot()

Availability

Fireworks 3

Description

Creates one or more hotspots from the selection.

Arguments

hotspotType, {*whatIfMultipleSelected*}

- Acceptable values for *hotspotType* are "hotspot" and "slice".
- *whatIfMultipleSelected* is an optional string that specifies how to create hotspots if multiple items are selected. Acceptable values for *whatIfMultipleSelected* are "single" (creates a single hotspot that has the same bounding rectangle as the selection), "multiple" (creates one hotspot for each item), and "ask user" (displays a dialog box to let the user decide).
- If *whatIfMultipleSelected* is omitted or null, "ask user" is assumed.

Returns

Nothing.

Example

The following command adds a hotspot to the selected item. If more than one item is selected, Fireworks creates one hotspot for each item.

```
fw.getDocumentDOM().copyToHotspot("hotspot", "multiple");
```

dom.cropSelection()

Availability

Fireworks 3

Description

Crops the selection to the specified rectangle.

Arguments

boundingRectangle

boundingRectangle is a rectangle that specifies the bounds within which the selection should be cropped (see “Rectangle” on page 6).

Returns

Nothing.

dom.deleteAllInDocument()

Availability

Fireworks MX

Description

Deletes all the objects in the document.

Arguments

None.

Returns

Nothing.

dom.deleteFrames()

Availability

Fireworks 3

Description

Deletes one or more frames.

Arguments

`frameIndex, howMany`

- *`frameIndex`* is a zero-based integer that specifies the location at which to begin deleting frames. To specify the current frame, pass `-1`.
- *`howMany`* specifies how many frames to delete.

Returns

Nothing.

dom.deleteLayer()

Availability

Fireworks 3

Description

Deletes a layer.

Arguments

`layerIndex`

`layerIndex` is a zero-based integer that specifies the layer to be deleted. To specify the current layer, pass `-1`.

Returns

Nothing.

Example

The following command deletes the current layer.

```
fw.getDocumentDOM().deleteLayer(-1);
```

dom.deletePointOnPath()

Availability

Fireworks 4

Description

Deletes the specified point on the currently selected path. If the point is the only one on its contour, the entire contour is deleted. If the point is the only one in the path, the entire path is deleted. The specified point does not need to be selected.

Arguments

contourIndex, pointIndex

- *contourIndex* is a zero-based integer that specifies the contour that contains the point to be deleted. To specify the current contour, pass `-1`.
- *pointIndex* is a zero-based integer that specifies the point to be deleted. To specify the current point, pass `-1`.

Returns

Nothing.

Example

The following command deletes the currently selected point.

```
fw.getDocumentDOM().deletePointOnPath(-1, -1);
```

dom.deleteSelection()

Availability

Fireworks 3

Description

Deletes the selection, or the pixel selection if Fireworks is in bitmap mode.

Arguments

bFillDeletedArea

- *bFillDeletedArea* is ignored if Fireworks is not in bitmap mode.
- If Fireworks is in bitmap mode and *bFillDeletedArea* is `true`, the deleted pixels are filled with the current fill color. If `false`, the deleted pixels are filled to transparent.

Returns

Nothing.

Example

If Fireworks is not in bitmap mode, the following command deletes the selected items.
If Fireworks is in bitmap mode, the following command fills the selected items to transparent.

```
fw.getDocumentDOM().deleteSelection(false);
```

dom.deleteSymbol()

Availability

Fireworks 3

Description

Deletes the specified symbols from the library.

Arguments

`symbolName`

`symbolName` is the name of the symbol to delete from the library. If more than one symbol exists with this name, only the first symbol is deleted.

- To delete all the selected symbols from the library (not document), pass `null`.
- If the deleted symbols contain any active instances in the document, the instances are also deleted.

Returns

Nothing.

Example

The following command deletes the selected symbols from the library as well as any active instances from the document.

```
fw.getDocumentDOM().deleteSymbol(null);
```

dom.detachInstanceFromSymbol()

Availability

Fireworks 3

Description

Breaks the links between the selected instances and the owning symbols.

Arguments

None.

Returns

Nothing.

dom.detachTextFromPath()

Availability

Fireworks 3

Description

Splits the selected text-on-a-path items into its original text and path items.

Arguments

None.

Returns

Nothing.

dom.distribute()

Availability

Fireworks 3

Description

Distributes the selection along a vertical or horizontal dimension.

Arguments

`dimension`
Acceptable values for *`dimension`* are `"vertical"` and `"horizontal"`.

Returns

Nothing.

dom.distributeLayerToFrames()

Availability

Fireworks 3

Description

Distributes the items on the specified layer to the frames of the document, adding frames if necessary. The first item on the layer goes to the first frame, the second item to the second frame, and so on. New frames are added to the document, if necessary. If there is only one item in the specified layer, this function has no effect.

Arguments

`layerIndex`
`layerIndex` is a zero-based integer that specifies the layer that contains the items to be distributed. To specify the current layer, pass `-1`.

Returns

Nothing.

dom.distributeSelectionToFrames()

Availability

Fireworks 3

Description

Distributes the selected items to the frames of the document, adding frames if necessary. The first item goes to the current frame, the second item to the next frame, and so on. If only one item is selected, this function has no effect.

Arguments

None.

Returns

Nothing.

dom.duplicateFrame()

Availability

Fireworks 3

Description

Duplicates a frame.

Arguments

`frameIndex`, *`howMany`*, *`where`*, *`bDupeSelectionOnly`*

- *`frameIndex`* is a zero-based integer that specifies the frame to duplicate. To specify the current frame, pass `-1`.
- *`howMany`* is an integer that specifies how many copies of the frame to make.
- Acceptable values for *`where`* are `"beginning"`, `"before current"`, `"after current"`, and `"end"`.
- If *`bDupeSelectionOnly`* is `true`, only items in the specified frame that are selected are duplicated to the new frame.

Returns

Nothing.

Example

The following command makes one copy of the current frame and places the new frame after the current frame.

```
fw.getDocumentDOM().duplicateFrame(-1, 1, "after current", false);
```

dom.duplicateLayer()

Availability

Fireworks 3

Description

Duplicates a layer.

Arguments

`layerIndex`, {*`howMany`*}, {*`where`*}

- *`layerIndex`* is a zero-based integer that specifies the layer to duplicate. To specify the current layer, pass `-1`.
- *`howMany`* is an optional integer that specifies how many times to duplicate the layer. If omitted, the layer is duplicated once.
- *`where`* is an optional argument that specifies where to put the new layer(s) in relation to the source layer. Acceptable values are `"beginning", "before current", "after current",` and `"end"`. If omitted, `"before current"` is assumed.

Returns

Nothing.

Example

The following command places three copies of the current layer at the end of the document.

```
fw.getDocumentDOM().duplicateLayer(-1, 3, "end");
```

dom.duplicateSelection()

Availability

Fireworks 3

Description

Makes a duplicate of the selection, offsetting it slightly from the original.

Arguments

None.

Returns

Nothing.

Example

The following command duplicates the selected items.

```
fw.getDocumentDOM().duplicateSelection();
```

Related functions

`dom.cloneSelection()`

dom.duplicateSelectionToFrameRange()

Availability

Fireworks 3

Description

Duplicates the selection to a range of frames of the document.

Arguments

`frameIndexFirst, frameIndexLast`

`frameIndexFirst` and *`frameIndexLast`* are zero-based integers that specify the range of frames (inclusive) to which the items should be copied. To specify the current frame, pass `-1`.

- If both arguments are the same, duplicates are placed only on that frame.
- If the range includes the current frame, duplicates are not placed on that frame.

Returns

Nothing.

dom.duplicateSelectionToFrames()

Availability

Fireworks 3

Description

Duplicates the selection to specified frames of the document.

Arguments

`whichFrames`

- Acceptable values for *`whichFrames`* are `"all"`, `"previous"`, `"next"`, and `"end"`.
- Note that `"end"` means the last frame of the document; it does not add a new frame.

Returns

Nothing.

dom.duplicateSymbol()

Availability

Fireworks 3

Description

Duplicates the specified symbol.

Arguments

`symbol`

`symbol` is the symbol to duplicate.

- To duplicate all selected symbols in the library (not the document), pass a `null` value.
- Duplicating a linked symbol results in a nonlinked duplicate.

Returns

Nothing.

dom.duplicateSymbolForAlias()

Availability

Fireworks 3

Description

If any symbol instances are selected, this function makes duplicate symbols of all the symbols that are pointed to by those instances. The selected instances are updated to point to the new duplicate copies of the symbols. Duplicate symbols always result in nonlinked duplicates. (The use of the word "alias" in the function name corresponds to an "instance" in a Fireworks document.)

Arguments

None.

Returns

Nothing.

dom.enableElementMask()

Availability

Fireworks 4, updated with new arguments in Fireworks MX

Description

Enables or disables the element mask on the selected element. Only one element can be selected when calling this function. If selecting more than one element (or none) at the time this function is called, Fireworks throws an exception.

Arguments

`enable`, *`selectAndEnterPaintModeIfPossible`*, *`newSelectionMask`*

- *`enable`* is a Boolean value that toggles the element mask between enabled (`true`) and disabled (`false`).
- *`selectAndEnterPaintModeIfPossible`* is a Boolean value that determines the mode for the mask. If *`selectAndEnterPaintModeIfPossible`* is `true`, and the mask is a bitmap mask, then bitmap mode is entered for the mask. It is `false` by default.
- *`newSelectionMask`* is an optional bitmap selection mask. If *`newSelectionMask`* is not `null`, and *`selectAndEnterPaintModeIfPossible`* is `true`, the selection will be set on the mask after entering paint mode. *`newSelectionMask`* is `null` by default.

Returns

Nothing.

dom.enableTextAntiAliasing()

Availability

Fireworks MX

Description

Turns anti-aliasing on or off for the selected blocks of text.

Note: To set the level of anti-aliasing, call "dom.setTextAntiAliasing()" on page 159.

Arguments

`antiAlias`

`antiAlias` is a Boolean value to turn anti-aliasing on (`true`) or off (`false`).

Returns

Nothing.

dom.enterElementMaskEditMode()

Availability

Fireworks 4

Description

Places Fireworks in element-mask edit mode for the selection. If the selection contains no mask elements, Fireworks throws an exception.

Arguments

None.

Returns

Nothing.

dom.enterPaintMode()

Availability

Fireworks 3, with the argument `newSelectionMask` added in Fireworks MX.

Description

Enters image edit mode on the selected items. Has no effect if nothing is selected or if a nonimage item is selected.

Arguments

`newSelectionMask`

`newSelectionMask` is an optional bitmap selection mask. When *`newSelectionMask`* is not null, the selection is set on the currently selected bitmap after entering paint mode. *`newSelectionMask`* is `null` by default.

Returns

Nothing.

dom.exitElementMaskEditMode()

Availability

Fireworks 4

Description

Takes Fireworks out of element-mask edit mode. If Fireworks is not in this mode, this function has no effect.

Arguments

None.

Returns

Nothing.

dom.exitPaintMode()

Availability

Fireworks 3

Description

Leaves bitmap mode. Has no effect if Fireworks is not in bitmap mode.

Arguments

None.

Returns

Nothing.

dom.exportOptions.loadColorPalette()

Availability

Fireworks 3

Description

Replaces the values in `dom.exportOptions.paletteEntries` with those in the specified GIF or ACT file. This function also sets `dom.exportOptions.paletteMode` to `"custom"`. For more information, see “ExportOptions” on page 33.

Arguments

`fileURL`

`fileURL` is a string, which is expressed as a file://URL, that specifies the GIF or ACT file that is used to replace the color panel.

Returns

`true` if the file is read successfully; `false` if the file is not the expected format or is not read successfully for any other reason.

dom.exportOptions.saveColorPalette()

Availability

Fireworks 3

Description

Saves the values in `dom.exportOptions.paletteEntries` to the specified color panel (ACT file). This function does not modify the document. For more information, see “ExportOptions” on page 33.

Arguments

`fileURL`

`fileURL` is a string, which is expressed as a file://URL, that specifies the name of the file to which the color panel should be saved. Do not specify a file extension; the .act extension is added automatically.

Returns

Nothing.

dom.exportTo()

Availability

Fireworks 3

Description

Exports the document as specified.

Arguments

`fileURL, {exportOptions}`

- *`fileURL`* is a string, which is expressed as a file://URL, that specifies the name of the exported file.
- *`exportOptions`* (optional) is an `ExportOptions` object (see "ExportOptions" on page 33). If this argument is omitted or `null`, the document's current Export Options settings are used. If values are passed in with *`exportOptions`*, they are used for this export operation only; they do not change the document's `exportOptions` property.

Returns

`true` if the file is successfully exported; `false` otherwise.

dom.fillSelectedPixels()

Availability

Fireworks 3

Description

When the selection is an image and Fireworks is in bitmap mode, this method fills the selected pixels with the current fill or generates a new pixel selection.

Arguments

`clickPt, p1, p2, p3, bFillSelectionOnly, tolerance, edgemode, featherAmt`

- *`clickPt`* is a point that specifies the *x,y* coordinates of the pixel to be filled or generated (see "Point" on page 6).
- *`p1`*, *`p2`*, and *`p3`* are points that specify the fill-vector. These arguments are ignored if the current fill does not use a fill-vector.
- If *`bFillSelectionOnly`* is `true`, the remaining arguments are ignored. If it is `false`, the current pixel selection is ignored, and a new one is generated using the values passed for *`tolerance`*, *`edgemode`*, and *`featherAmt`*. (This behavior is the same as if the Magic Wand tool were used at the *`clickPt`* location.)
- *`tolerance`* is an integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- Acceptable values for *`edgemode`* are `"hard edge"`, `"antialias"`, and `"feather"`.
- *`featherAmt`* is an integer between 0 and 32,000, inclusive, that specifies the number of pixels to feather. This value is ignored if *`edgemode`* is not `"feather"`.

Returns

Nothing.

Example

The following command fills the selection with a hard edge, and the tolerance set to 32.

```
fw.getDocumentDOM().fillSelectedPixels({x:207, y:199}, {x:207, y:199}, {x:207,
  y:199}, {x:207, y:199}, false, 32, "hard edge", 0);
```

dom.filterSelection()

Availability

Fireworks 3

Description

Applies the specified pixel filter to the selection. Nonimage items are converted into images before the filter is applied. Only external filters that are capable of also being Live Effects can be applied using this function. To apply other types of external filters, use `dom.filterSelectionByName()`.

Arguments

`LiveEffect`

`LiveEffect` is an `Effect` object (see “Effect” on page 23).

Returns

Nothing.

Example

The following command runs the selected pixels through the hue/saturation filter and then sets hue to 30 and saturation to 20.

```
fw.getDocumentDOM().filterSelection({
  EffectMoaID:"{3439b08d-1922-11d3-9bde00e02910d580}",
  hls_colorize:true, hue_amount:30, lightness_amount:0, saturation_amount:20
});
```

dom.filterSelectionByName()

Availability

Fireworks 3

Description

Applies the specified pixel filter to the selection as a permanent action, not as a Live Effect. (To apply filters that can also be Live Effects, you can use `dom.filterSelection()`.) This function always displays a dialog box.

Arguments

`category, name`

- *`category`* is a string that specifies the category of the pixel filter to be applied. Acceptable values depend on which filters you have installed.
- *`name`* is a string that specifies the name of the pixel filter to be applied. Acceptable values depend on which filters you have installed.

Returns

Nothing.

dom.findExportFormatOptionsByName()

Availability

Fireworks 3

Description

Looks for a set of export settings that were saved with the specified name.

Arguments

`name`

`name` is a string that specifies the name of the set of export settings to find.

Returns

If there is a set of export settings with the specified name, the argument returns an object that represents it; otherwise, it returns `null`.

dom.findNamedElements()

Availability

Fireworks 4

Description

Looks for elements that have the specified name.

Arguments

`name`

`name` is a case-sensitive string that specifies the exact element name to find. To specify elements that have no name, pass `null`.

Returns

An array of elements that have the specified name, or `null` if no objects have the specified name.

Related functions

`dom.setElementName()`

dom.flattenDocument()

Availability

Fireworks 3

Description

Flattens the entire document into a single pixel image. This is the same behavior as the Merge Layers command.

Arguments

None.

Returns

Nothing.

dom.flattenSelection()

Availability

Fireworks 3

Description

Flattens the selection into a single pixel image. This action is the same behavior as the Merge Images command.

Arguments

None.

Returns

Nothing.

dom.getFontMarkup()

Availability

Fireworks 3

Description

Gets a font markup attribute for the selected text.

Arguments

`fontAttribute`
Acceptable values for *`fontAttribute`* are `"size"`, `"color"`, and `"face"`.

Returns

A string that specifies the markup value. Returns `null` if the text has multiple attributes or if the selection contains no text.

dom.getPixelMask()

Availability

Fireworks 3, deprecated in 4

Description

Gets the current pixel-selection mask. The result of this call could be used to call "dom.enableElementMask()" on page 88 or "dom.enterPaintMode()" on page 90.

Arguments

None.

Returns

The mask for the current pixel selection. Returns `null` if Fireworks is not in bitmap mode, or if there is no pixel selection. For information on the format of mask variables, see "Mask" on page 6.

dom.getSelectionBounds()

Availability

Fireworks 3

Description

Gets the bounding rectangle of the selection.

Arguments

None.

Returns

A rectangle (see "Rectangle" on page 6). Returns `null` if nothing is selected.

dom.getShowGrid()

Availability

Fireworks 3

Description

Determines if the grid is visible.

Arguments

None.

Returns

`true` if the grid is visible; `false` otherwise.

dom.getShowRulers()

Availability

Fireworks 3

Description

Determines if the rulers are visible.

Arguments

None.

Returns

`true` if the rulers are visible; `false` otherwise.

dom.getSnapToGrid()

Availability

Fireworks 3

Description

Determines if the Snap to Grid function is active.

Arguments

None.

Returns

`true` if the Snap to Grid function is active; `false` otherwise.

dom.getTextAlignment()

Availability

Fireworks 3

Description

Gets the alignment of selected text.

Arguments

None.

Returns

One of the following strings: `"left"`, `"center"`, `"right"`, `"justify"`, `"stretch"`, `"vertical left"`, `"vertical center"`, `"vertical right"`, `"vertical justify"`, or `"vertical stretch"`. Returns `null` if the text has multiple alignments or if the selection contains no text.

dom.group()

Availability

Fireworks 3, argument deprecated in 4

Description

Groups the selection. To ungroup elements use `dom.ungroup()` (see "dom.ungroup()" on page 168).

Arguments

`{`*`type`*`}`

`type` is an optional string that specifies how to group the items. Acceptable values are `"normal"`, `"mask to image"`, and `"mask to path"`. If the argument is omitted, `"normal"` is assumed. `"mask to image"` and `"mask to path"` are deprecated in Fireworks 4.

Returns

Nothing.

Example

The following command sets the selected group to mask to the image.

```
replace with fw.getDocumentDOM().group("normal");
```

dom.hasCharacterMarkup()

Availability

Fireworks 3, enhanced in 4

Description

Determines if the selected text has the specified character markup.

Arguments

`tag`

Acceptable values for *`tag`* are `"b"`, `"i"`, and `"u"`, for bold, italic, and underline; and `"fwplain"`, which was added in Fireworks 4, for text without character markup.

Returns

`true` if the text has the specified character markup; `false` if it does not or if only part of the text has the markup.

dom.hideSelection()

Availability

Fireworks 3

Description

Hides the selection. To redisplay it, use `dom.showAllHidden()`.

Arguments

None.

Returns

Nothing.

dom.importFile()

Availability

Fireworks 3

Description

Imports the specified file at the specified location.

Arguments

`fileURL`, *`boundingRectangle`*, *`bMaintainAspectRatio`*

- *`fileURL`* is the filename of the file to be imported, which is expressed as a file://URL.
- *`boundingRectangle`* is a rectangle that specifies the size to make the imported file (see "Rectangle" on page 6). If *`boundingRectangle`* is specified with `left == right` and `top == bottom`, the file is brought in unscaled with its top-left corner at the specified location, and the third argument is ignored.
- *`bMaintainAspectRatio`* is `true`, the file is scaled to the largest size that fits within *`boundingRectangle`* while retaining the file's current aspect ratio. (This is a handy option for creating thumbnails.) If it is `false`, the file is scaled to fill *`boundingRectangle`*.

Returns

Nothing.

Example

The following command imports the specified file and maintains its aspect ratio.

```
fw.getDocumentDOM().importFile("file:///C|/images/foo.psd", {left:25, top:50,
  right:100, bottom:250}, true);
```

dom.importSymbol()

Availability

Fireworks 3

Description

Imports the specified external graphics file (for example, GIF, JPEG, or Fireworks document) into the library of the document.

Arguments

`fileURL, bAddToDoc, bAllowUI`

- *`fileURL`* is the name of the file to be imported into the library, which is expressed as a file:// URL.
- If *`bAddToDoc`* is `true`, the symbol is added to the library and an instance of the symbol is inserted into the center of the document. If it is `false`, the symbol is added only to the library.
- If *`bAllowUI`* is `true`, and *`fileURL`* is a Fireworks document that contains symbols, then a dialog box lets the user specify which symbols to import from the external file. If it is `false`, all the symbols in the external file are imported.

Returns

Nothing.

dom.importSymbolButNotAsAlias()

Availability

Fireworks MX

Description

`dom.importSymbolButNotAsAlias` extracts the component elements from the selected symbol and places copies of those elements in the document.

This function is similar to the `dom.importSymbol` API. `dom.importSymbol` places an instance of a symbol in your document—for example, when you select Edit > Libraries > Buttons, and `dom.importSymbolButNotAsAlias` extracts the component elements from the selected symbol and places copies of those elements in the document. `dom.importSymbolButNotAsAlias` does not place in an instance in the document.

Arguments

`filepath`, *`whichSymbol`*

- *`filepath`* is the *`fileURL`* of the file that contains the symbol to be copied.
- *`whichSymbol`* is the index of the symbol within the document, which is specified in the *`filepath`*.

Returns

Nothing.

dom.inLaunchAndEdit()

Availability

Fireworks MX

Description

Identifies if the document was opened by a Launch and Edit operation.

Arguments

None.

Returns

A Boolean value: `true` if opened by a Launch and Edit operation; `false` otherwise.

dom.insertPointInPath()

Availability

Fireworks 3

Description

Inserts a Bézier point in the selected path. This function is similar to `dom.appendPointToPath()` but includes a *`tParameter`* argument, which lets you control where the point is inserted.

Arguments

`contourIndex, ptToInsertBefore, tParameter, controlPointFirst, mainPoint, controlPointLast`

- *`contourIndex`* is a zero-based integer that specifies the contour into which the Bézier point is inserted. For paths with multiple contours, the contours are in an arbitrary order.
- *`ptToInsertBefore`* is a zero-based integer that specifies where the new point should be placed on the path. The new point is appended in front of the point that this integer represents: To add a point to the beginning of the path, pass `0`; to add a point to the end of the path, pass a large number.
- *`tParameter`* is a float value between 0 and 1 that specifies where to insert the new point in the Bézier segment.
- *`controlPointFirst`*, *`mainPoint`*, and *`controlPointLast`* are points that specify the *x,y* coordinates of the preceding control point, the main point, and the following control point of the new point (see "Point" on page 6).

Returns

Nothing.

`Related Functions  dom.appendPointToPath()`

dom.isSelectionDirectlyAboveBitmapObject()

Availability

Fireworks MX

Description

Tests to see if the selected object(s) are directly above a bitmap object. The selection does not need to be contiguous, although at least one item in the selection must be directly above a bitmap.

Arguments

None.

Returns

A Boolean value: `true` if the selected objects are directly above an image element; `false` otherwise.

dom.joinPaths()

Availability

Fireworks 3

Description

Joins the selected paths.

Arguments

None.

Returns

Nothing.

dom.knifeElementsFromPoint()

Availability

Fireworks 3

Description

When the user clicks a single point while using the Knife tool, this function cuts additional items within the specified tolerance. This action is similar to using the Knife tool with a single click.

Arguments

`from, tolerance`

- *`from`* is a point that specifies the *x,y* coordinates of the point that the user clicked (see "Point" on page 6).
- *`tolerance`* is a float value > = 0 that specifies the tolerance within which items are cut.

Returns

`true` if anything was cut; `false` otherwise.

Related functions

`dom.knifeElementsFromPoints()`

dom.knifeElementsFromPoints()

Availability

Fireworks 3

Description

When the user drags while using the Knife tool, this function cuts additional items within the specified tolerance. This action is similar to using the Knife tool with a Drag operation.

Arguments

`from, to, tolerance`

- *`from`* is a point that specifies the *x,y* coordinates of the point where the user clicked and started to drag (see "Point" on page 6).
- *`to`* is a point that specifies the *x,y* coordinates of the point where the user ended the Drag operation.
- *`tolerance`* is a float value > = 0 that specifies the tolerance within which items are cut.

Returns

`true` if anything is cut; `false` otherwise.

Related functions

`dom.knifeElementsFromPoint()`

dom.linkElementMask()

Availability

Fireworks 4

Description

Links or unlinks the element mask on the selected element. Only one element can be selected when calling this function. If selecting more than one element (or none) at the time this function is called, Fireworks throws an exception. An exception is also thrown if the element has no element mask.

Arguments

`frame, layer, element, bLink`

- *`frame`* is a zero-based integer that specifies the frame that contains the element. To specify the current frame, pass –1.
- *`layer`* is a zero-based integer that specifies the layer that contains the element. To specify the current layer, pass –1.
- *`element`* is a zero-based integer that specifies the element. To specify the current element, pass –1.
- If *`bLink`* is `true`, the element masks are linked to their elements; if `false`, they are unlinked from their elements.

Returns

Nothing.

dom.makeFind()

Availability

Fireworks 3

Description

Creates an object of class `Find` to perform a find-and-replace operation in this document.

Arguments

`findSpec`

`findSpec` is a `Find` object (see “Find” on page 15).

dom.makeGoodNativeFilePath()

Availability

Fireworks 3

Description

Ensures that the specified file URL ends in a .png extension. Does not affect the name of the file on disk.

Arguments

`fileURL`

`fileURL` is the name of the file, which is expressed as a file://URL, whose extension should be changed to .png.

Returns

A string that contains the file URL with a .png extension.

Example

The following command returns `"file:///My Documents/image01.png"`.

```
fw.getDocumentDOM().makeGoodNativeFilePath("file:///My Documents/image01.png")
```

dom.makeActive()

Availability

Fireworks 3

Description

Makes the selected document active for editing.

Arguments

None.

Returns

Nothing.

dom.mergeDown()

Availability

Fireworks MX

Description

Merges selected objects to the bitmap directly below the selected objects. Succeeds only if the object immediately below the selection is a bitmap. See "dom.isSelectionDirectlyAboveBitmapObject()" on page 100.

Arguments

None.

Returns

Nothing.

dom.modifyPointOnPath()

Availability

Fireworks 3

Description

Modifies an existing point on the selected path.

Arguments

`contourIndex`, *`ptToModify`*, *`controlPointFirst`*, *`mainPoint`*, *`controlPointLast`*, *`dReapplyAttrs`*, *`bClosePath`*

- *`contourIndex`* is a zero-based integer that specifies the contour into which the Bézier point is inserted. For paths with multiple contours, the contours are in an arbitrary order.
- *`ptToModify`* is a zero-based integer that specifies the point to be modified.

- `controlPointFirst`, `mainPoint`, and `controlPointLast` are points that specify the *x,y* coordinates of the preceding control point, the main point, and the following control point of the new point (see “Point” on page 6).
- If `dReapplyAttrs` is `true`, the path has the document’s current fill, stroke, and so on reapplied to it. If it is `false`, the path attributes are not changed.
- If `bClosePath` is `true`, the path is marked as closed after modifying the point. If it is `false`, the path retains its original open or closed value.

Returns

Nothing.

dom.moveBezierHandleBy()

Availability

Fireworks 3

Description

Moves the specified point’s Bézier handles by a certain amount.

Arguments

`whichPath, contourIndex, ptToModify, deltaControlPointFirst, deltaControlPointLast`

- `whichPath` is a zero-based integer that specifies an index into the list of selected items, indicating which item contains the Bézier handles to move.
- `contourIndex` is a zero-based integer that specifies the contour that contains the handles to move. For paths with multiple contours, the contours are in an arbitrary order.
- `ptToModify` is a zero-based integer that specifies the point whose handles are moved.
- `deltaControlPointFirst` and `deltaControlPointLast` are points that specify the *x,y* coordinate values by which the preceding control point and the following control point of *ptToModify* are moved. For example, passing (`{x:1,y:2}`) specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.moveElementMaskBy()

Availability

Fireworks 4

Description

For all the elements in the selection that have element masks (linked or unlinked), it moves the element masks by the specified amount. Elements without element masks are ignored. If no elements in the selection have element masks, an exception is thrown.

Arguments

`delta`

`delta` is a point that specifies the *x,y* coordinate values by which the element masks are moved (see "Point" on page 6). For example, passing ({*x*:`1`,*y*:`2`}) specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.moveFillVectorHandleBy()

Availability

Fireworks 3

Description

If the selection has a fill that uses a fill vector (for example, a gradient fill), this function adjusts the handles of the fill vector. If the selection does not, this function has no effect.

Arguments

`delta, whichHandle, bConstrain, bMoveJustOne`

- *`delta`* is a point that specifies the *x,y* coordinate values by which the handle is moved (see "Point" on page 6). For example, passing `({`*x*:`1`,*y*:`2})` specifies a location that is right by one pixel and down by two pixels.
- *`whichHandle`* specifies which handle to move and can be one of the following values: `"start"`, `"end1"`, `"end2"`, `"rotate1"`, or `"rotate2"`. (Some fills ignore `"end2"`.) Use `"rotate1"` or `"rotate2"` to rotate the `end1` or `end2` point around the start point.
- If *`bConstrain`* is `true`, movement is constrained to 45-degree increments.
- If *`bMoveJustOne`* is `true`, only the specified handle moves. If it is `false`, other handles might move in sync when the specified handle is moved.

Returns

Nothing.

dom.moveMaskGroupContentsBy()

Availability

Fireworks 3

Description

If the selection is a mask group, this function moves the contents within the mask group by the specified amount. If the selected element has an element mask, this function moves the element (not the element mask) by the specified amount.

Arguments

`delta`

`delta` is a point that specifies the *x,y* coordinate values by which the element is moved (see "Point" on page 6). For example, passing ({*x*:`1`,*y*:`2`}) specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

Related functions

`dom.moveElementMaskBy()`

dom.movePixelMaskBy()

Availability

Fireworks 4

Description

Moves a bitmap mode selection by the specified amount, without moving the pixels that are within the selection.

Arguments

`delta`

`delta` is a point that specifies the *x,y* coordinate values by which the bitmap mode selection is moved (see “Point” on page 6). For example, passing (`{`*`x`*`:1,`*`y`*`:2}`) specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.movePointOnHotspotBy()

Availability

Fireworks 3

Description

If the selection is a hotspot or slice of the polyline variety, this function moves a point on the hotspot’s path by the specified amount.

Arguments

`ptToModifyIndex, delta`

- *`ptToModifyIndex`* is a zero-based integer that specifies which point on the path is to move.
- *`delta`* is a point that specifies the *x,y* coordinate values by which the point is moved (see “Point” on page 6). For example, passing `({`*`x`*`:1,`*`y`*`:2})` specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.movePointOnHotspotByWithFlags()

Availability

Fireworks MX

Description

If the selection is a hotspot or slice of the polyline variety, this function moves a point on the hotspot's path by the specified amount.

Arguments

`ptToModifyIndex`, `delta`, `flags`

`ptToModifyIndex` is a zero-based integer that specifies which point on the path is to move.

`delta` is a point that specifies the *x,y* coordinate values by which the point is moved (see "Point" on page 6). For example, passing `({x:1,y:2})` specifies a location that is right by one pixel and down by two pixels.

`flags` is a Boolean value that determines whether this slice or hotspot will be duplicated. This argument is important for giving slices a unique name so their behaviors remain unaffected.

Returns

Nothing.

dom.moveSelectedBézierPointsBy()

Availability

Fireworks 3

Description

If the selection contains at least one path with at least one Bézier point selected, this function moves all selected Bézier points on all selected paths by the specified amount.

Arguments

`delta`
`delta` is a point that specifies the *x,y* coordinate values by which the selected Bézier points are moved (see "Point" on page 6). For example, passing `({x:1,y:2})` specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.moveSelectionBy()

Availability

Fireworks 3

Description

Moves the selected items by the specified amount or makes a copy of them and offsets them from the original by the specified amount.

Arguments

`delta, bMakeCopy, doSubSel`

- *`delta`* is a point that specifies the *x,y* coordinate values by which the selection moved (see "Point" on page 6). For example, passing `({`*x*`:1,`*y*`:2})` specifies a location right by one pixel and down by two.
- *`bMakeCopy`* are the items that are copied instead of moved.
- If *`doSubSel`* is set to `true` the function moves only the subselected parts of a path. If the argument is set to `false`, the function moves the whole object.

Returns

Nothing.

Example

The following command moves the selected items right by 62 pixels and down by 84 pixels.

```
fw.getDocumentDOM().moveSelectionBy({x:62, y:84}, false, false);
```

dom.moveSelectionMaskBy()

Availability

Fireworks 4

Description

Moves the current pixel mask by the specified amount. If there is no pixel selection, an exception is thrown.

Arguments

`delta`

`delta` is a point that specifies the *x,y* coordinate values by which the mask is moved (see "Point" on page 6). For example, passing `({`*x*`:1,`*y*`:2})` specifies a location that is right by one pixel and down by two pixels.

Returns

Nothing.

dom.moveSelectionTo()

Availability

Fireworks 3

Description

Moves or copies the selection to the specified location.

Arguments

`location, bMakeCopy, doSubSel`

- *`location`* is a point that specifies the *x,y* coordinate values of the location to which the selection is moved or copied (see "Point" on page 6).
- *`bMakeCopy`* specifies copying instead of moving the selection.
- *`doSubSel`* is set to `true` the function moves only the subselected parts of a path. If the argument is set to `false`, the function moves the whole object.

Returns

Nothing.

Example

The following command copies only the selected parts of a path to the specified coordinates:

```
fw.getDocumentDOM().moveSelectionTo({x:163, y:0}, true, true);
```

dom.moveSelectionToFrame()

Availability

Fireworks 3

Description

Moves or copies the selection to the specified frame.

Arguments

`frameIndex, bMakeCopy`

- *`frameIndex`* is a zero-based integer that specifies the frame to which the selection is moved or copied. To specify the current frame, pass `-1`.
- If *`bMakeCopy`* is `true`, the selection is copied instead of moved.

Returns

Nothing.

dom.moveSelectionToLayer()

Availability

Fireworks 3, enhanced in 4

Description

Moves or copies the selection to the specified layer.

Arguments

`layerIndex, bMakeCopy, {whatIfMultipleSelected}, {elementIndex}`

- *`layerIndex`* is a zero-based integer that specifies the layer to which the selection should be moved or copied. To specify the current layer, pass `-1`.
- If *`bMakeCopy`* is `true`, the selection is copied instead of moved.
- *`whatIfMultipleSelected`* is an optional string that is used only if the destination is a web layer and *`bMakeCopy`* is `true`. It specifies how to create hotspots if multiple items are selected. Acceptable values for *`whatIfMultipleSelected`* are `"single"` (creates a single hotspot that has the same bounding rectangle as the selection), `"multiple"` (creates one hotspot for each item), and `"ask user"` (displays a dialog box to let the user decide). If *`whatIfMultipleSelected`* is omitted or `null`, `"ask user"` is assumed.
- *`elementIndex`*, which was added in Fireworks 4, is a zero-based index that specifies the element before which the moved or copied selection should be inserted. If *`elementIndex`* is omitted, the selection is placed at the top of the layer (before any other elements). Otherwise, it is an index within the existing elements in the layer, where `0` is the topmost, and (`n-1`) is the last element (for a layer with *n* elements). The maximum value is the number of elements previously in the layer—meaning that the elements are moved to the bottom of the specified layer.

Returns

Nothing.

dom.moveSelectionToNewLayer()

Availability

Fireworks 3

Description

Makes a new layer with a default name, then moves or copies the selection to that new layer.

Arguments

`bMakeCopy`
If *`bMakeCopy`* is `true`, the selected items are copied instead of moved.

Returns

Nothing.

dom.pathCrop()

Availability

Fireworks 3

Description

Performs a Crop operation on the selected paths.

Arguments

None.

Returns

Nothing.

dom.pathExpand()

Availability

Fireworks 3

Description

Performs an Expand operation on the selected paths.

Arguments

`width, miter, cap, join`

- *`width`* is a float value that specifies the new width of the selected paths, in pixels.
- *`miter`* is a float value that specifies the new miter angle of the selected paths, in pixels. This argument is ignored if *join* is not `"miter"`.
- Acceptable values for *`cap`* are `"butt"`, `"square"`, and `"round"`.
- Acceptable values for *`join`* are `"bevel"`, `"round"`, and `"miter"`.

Returns

Nothing.

dom.pathInset()

Availability

Fireworks 3

Description

Performs an Inset operation on the selected paths.

Arguments

`width, miter, join`

- *`width`* is a float value that specifies the new width of the selected paths, in pixels.
- *`miter`* is a float value that specifies the new miter angle of the selected paths, in pixels. This argument is ignored if *`join`* is not `"miter"`.
- Acceptable values for *`join`* are `"bevel"`, `"round"`, and `"miter"`.

Returns

Nothing.

dom.pathIntersect()

Availability

Fireworks 3

Description

Performs an Intersect operation on the selected paths.

Arguments

None.

Returns

Nothing.

dom.pathPunch()

Availability

Fireworks 3

Description

Performs a Punch operation on the selected paths.

Arguments

None.

Returns

Nothing.

dom.pathSimplify()

Availability

Fireworks 3

Description

Performs a Simplify operation on the selected paths.

Arguments

`limit`

`limit` is a float value that specifies how much to simplify. This value corresponds to the value in the Modify > Alter Path > Simplify dialog box.

Returns

Nothing.

dom.pathUnion()

Availability

Fireworks 3

Description

Performs a Union operation on the selected paths.

Arguments

None.

Returns

Nothing.

dom.previewInBrowser()

Availability

Fireworks MX

Description

Previews the document in the primary or secondary browser.

Arguments

`primaryBrowser`

`primaryBrowser` is a Boolean value that specifies which browser, primary (`true`), or secondary (false), should be launched by Fireworks.

Returns

Nothing.

dom.rebuildColorTable()

Availability

Fireworks 3

Description

Rebuilds the color table for the current export settings of the document. This is the same behavior as choosing Rebuild Color Table from the Color Table panel.

Arguments

None.

Returns

Nothing.

dom.redo()

Availability

Fireworks 3

Description

Reinstates the last action that was undone in the document.

Arguments

None.

Returns

Nothing.

dom.redraw()

Availability

Fireworks MX

Description

Forces the document to redraw itself immediately. This function is useful for providing feedback during complicated commands.

Arguments

None.

Returns

Nothing.

dom.reflectSelection()

Availability

Fireworks 3

Description

Reflects the selection vertically, horizontally, or both.

Arguments

`bHoriz, bVert, opts`

- If *`bHoriz`* is `true`, the items reflect horizontally.
- If *`bVert`* is `true`, the items reflect vertically.
- Acceptable values for *`opts`* are `"transformAttributes"`, `"autoTrimImages"`, and `"autoTrimImages transformAttributes"`.

Returns

Nothing.

dom.removeAllGuides()

Availability

Fireworks 3

Description

Removes all guides of the specified type.

Arguments

`guidekind`

Acceptable values for *`guidekind`* are `"horizontal"` and `"vertical"`.

Returns

Nothing.

dom.removeBehavior()

Availability

Fireworks 3

Description

Removes one or all behavior events from the selected hotspots and slices.

Arguments

`{`*`event`*`}, {`*`eventIndex`*`}`

- *`event`* and *`eventIndex`* are optional; if they are omitted, this function removes all events from selected hotspots and slices.
- *`event`* specifies the event that triggers the behavior. This argument is ignored by Fireworks.
- *`eventIndex`* is a zero-based integer that specifies the location of the behavior to be removed. To specify the end location, pass `-1` here.

Returns

Nothing.

`Related Functions  dom.addBehavior()`

dom.removeBrush()

Availability

Fireworks 3

Description

Sets the brush of the selection to None.

Arguments

None.

Returns

Nothing.

dom.removeCharacterMarkup()

Availability

Fireworks 3

Description

Reapplies the default value for the specified markup type to the text in the selection.

Arguments

`tag`

Acceptable values for *`tag`* are `"b"`, `"i"`, and `"u"`, for bold, italic, and underline.

Returns

Nothing.

dom.removeElementMask()

Availability

Fireworks 4

Description

Removes the mask from the selected element. Only one element can be selected when calling this function. If selecting more than one element (or none) at the time this function is called, Fireworks throws an exception.

Arguments

`whatIfElementIsAnImage`

- *`whatIfElementIsAnImage`* is used only if the element (not the element mask) is an image. Acceptable values for *`whatIfElementIsAnImage`* are `"apply"` (apply the element mask to the image before discarding the element mask), `"discard"` (discard the element mask), and `"ask"` (displays a dialog box to let the user decide).
- If you pass `"ask"` and the user cancels the dialog box, Fireworks returns an error.

Returns

Nothing.

dom.removeFontMarkup()

Availability

Fireworks 3

Description

Reapplies the default value for the specified font attribute to the text in the selection.

Arguments

`fontAttribute`

Acceptable values for *`fontAttribute`* are `"size"`, `"color"`, and `"face"`.

Returns

Nothing.

dom.removeFill()

Availability

Fireworks 3

Description

Sets the fill of the selection to None.

Arguments

None.

Returns

Nothing.

dom.removeGuide()

Availability

Fireworks 3

Description

Removes the specified guide. If no guide is at that position, this function has no effect.

Arguments

`position, guidekind`

- *`position`* is a float value that specifies the position of the guide to be removed.
- Acceptable values for *`guidekind`* are `"horizontal"` and `"vertical"`. If *`guidekind`* is `"horizontal"`, it is assumed that *`position`* is a *y* coordinate; if *`guidekind`* is `"vertical"`, it is assumed that *`position`* is an *x* coordinate.

Returns

Nothing.

dom.removeTransformation()

Availability

Fireworks 3

Description

Removes the transformations, if any, from the selected text or instances.

Arguments

None.

Returns

Nothing.

dom.reorderFrame()

Availability

Fireworks 3

Description

Moves or copies the specified frame before another specified frame.

Arguments

`frameToMove`, *`frameToPutItBefore`*, *`bMakeCopy`*

- *`frameToMove`* is a zero-based integer that specifies which frame to move or copy.
- *`frameToPutItBefore`* is a zero-based integer that specifies which frame you want to move or copy the frame before. That is, if you pass `1` for *`frameToMove`* and `0` for *`frameToPutItBefore`*, the second frame is placed before the first frame.
- If *`bMakeCopy`* is `true`, the specified frame is copied instead of moved.

Returns

Nothing.

Example

The following command moves the third frame before the first frame.

```
fw.getDocumentDOM().reorderFrame(2, 0, false);
```

dom.reorderLayer()

Availability

Fireworks 3

Description

Moves or copies the specified layer before another specified layer.

Arguments

`layerToMove`, *`layerToPutItBefore`*, *`bMakeCopy`*

- *`layerToMove`* is a zero-based integer that specifies which layer to move or copy.
- *`layerToPutItBefore`* is a zero-based integer that specifies which layer to move or copy the layer before. That is, if you pass `1` for *`layerToMove`* and `0` for *`layerToPutItBefore`*, the second layer is placed before the first layer.

- If *bMakeCopy* is `true`, the specified layer is copied instead of moved.

Returns

Nothing.

dom.replaceButtonTextStrings()

Availability

Fireworks 3

Description

Replaces all text items (selected and unselected) within the document that are defined as Button Text items with the specified string. (Button Text items are defined as the topmost text items on each frame.)

Arguments

newString, uniformAttrs

- *newString* specifies the string to be used as replacement text.
- If *uniformAttrs* is `false`, each character retains the attributes of the character that was formerly in its position; that is, Fireworks preserves the existing formatting. If *uniformAttrs* is `true`, all characters assume the attributes of the first character in the string that is being replaced.

Returns

Nothing.

Related functions

`dom.replaceButtonTextStringsInInstances()`

dom.replaceButtonTextStringsInInstances()

Availability

Fireworks 3

Description

Replaces selected button text items with the specified string. (Button text items are defined as the topmost text items on each frame.)

Arguments

newString, uniformAttrs

- *newString* specifies the string to be used as replacement text.
- If *uniformAttrs* is `false`, each character retains the attributes of the character that was formerly in its position; that is, Fireworks preserves the existing formatting. If *uniformAttrs* is `true`, all characters assume the attributes of the first character in the string that is being replaced.

Returns

Nothing.

Related functions

`dom.replaceButtonTextStrings()`

dom.replaceTextString()

Availability
Fireworks 3

Description
Replaces the text of all selected text items with the specified string.

Arguments
`newString, uniformAttrs`

- *`newString`* specifies the string to be used as replacement text.
- If *`uniformAttrs`* is `false`, each character retains the attributes of the character that was formerly in its position; that is, Fireworks preserves the existing formatting. If *`uniformAttrs`* is `true`, all characters assume the attributes of the first character in the string that is being replaced.

Returns
Nothing.

dom.resizeSelection()

Availability
Fireworks 3

Description
Resizes the selection to the specified pixel width and height, keeping the top-left corner of the selection in place.

Arguments
`width, height`

`width` and *`height`* are integers that specify the new width and height in pixels.

Returns
Nothing.

dom.restoreJPEGMask()

Availability
Fireworks 4

Description
Restores the selection that is specified in `dom.saveJPEGMask()` (see “dom.saveJPEGMask()” on page 122).

Arguments
None.

Returns
Nothing.

Related functions
`dom.saveJPEGMask()`

dom.restoreSelection()

Availability

Fireworks 4

Description

Restores the selection that is specified in `dom.saveSelection()` (see “dom.saveSelection()” on page 122).

Arguments

None.

Returns

Nothing.

Related functions

`dom.saveSelection()`

dom.reversePathTextDirection()

Availability

Fireworks 3

Description

For all text-on-a-path items in the selection, it reverses the direction of the text along the path.

Arguments

None.

Returns

Nothing.

dom.rotateDocument()

Availability

Fireworks 3

Description

Rotates the entire document 90, 180, or 270 degrees clockwise. Rotating 270 degrees is the same behavior as rotating 90 degrees counterclockwise.

Arguments

`rotationAmount`
Acceptable values for *`rotationAmount`* are 90, 180, and 270.

Returns

Nothing.

dom.rotateSelection()

Availability

Fireworks 3

Description

Rotates the selection clockwise by the specified number of degrees. Rotating 270 degrees is the same behavior as rotating 90 degrees counterclockwise.

Arguments

`rotationDegrees, opts`

- *`rotationDegrees`* is a float value that specifies the number of degrees to rotate the selection.
- Acceptable values for *`opts`* are `"transformAttributes"`, `"autoTrimImages"`, and `"autoTrimImages transformAttributes"`.

Returns

Nothing.

dom.save()

Availability

Fireworks 3

Description

Saves the document in its default location. After a successful Save operation, the document's `isDirty` property clears.

Arguments

`{`*`bOkToSaveAs`*`}`

If *`bOkToSaveAs`* is `true` or omitted and the file was never saved, then the Save As dialog box appears. If *`bOkToSaveAs`* is `false` and the file was never saved, the file is not saved.

Returns

`true` if the Save operation completes successfully; `false` otherwise.

dom.saveCopyAs()

Availability

Fireworks 3

Description

Saves a copy of the document in a specified directory with a specified name. This function does not affect the document's `filePathForSave` or `isDirty` properties.

Arguments

`fileURL`

`fileURL` is a string, which is expressed as a file://URL, that specifies the directory and name under which the copy should be saved.

Returns

`true` if the Save operation completes successfully; `false` otherwise.

dom.saveJPEGMask()

Availability

Fireworks 4

Description

Stores the current selection in bitmap mode as the "Selective JPEG mask". Use `dom.restoreJPEGMask()` to restore the mask (see "dom.restoreJPEGMask()" on page 119).

Arguments

None.

Returns

Nothing.

Related Functions

`dom.restoreJPEGMask()`

dom.saveSelection()

Availability

Fireworks 4

Description

Stores the current selection in bitmap mode as the saved selection. Use `dom.restoreSelection()` to restore the selection ("dom.restoreSelection()" on page 120).

Arguments

None.

Returns

Nothing.

Related functions

`dom.restoreSelection()`

dom.scaleSelection()

Availability

Fireworks 3

Description

Scales the selection in the horizontal and vertical axes.

Arguments

`xScaleAmount, yScaleAmount, opts`

- *`xScaleAmount`* and *`yScaleAmount`* are float values that specify the amount to scale the selection in the horizontal and vertical axes. Acceptable values are 0.0 or greater; a value of 1 represents 100 percent, 2 represents 200 percent, and so on.
- Acceptable values for *`opts`* are `"transformAttributes"`, `"autoTrimImages"`, and `"autoTrimImages transformAttributes"`.

Returns

Nothing.

Example

The following command scales the selected items to approximately two-thirds (67 percent) and automatically trims the images and transforms the attributes.

```
fw.getDocumentDOM().scaleSelection(0.67, 0.67, "autoTrimImages
  transformAttributes");
```

dom.selectAdjustPixelSel()

Availability

Fireworks 3

Description

Expands or reduces the pixel selection by the specified number of pixels, selects a border of pixels, or smooths the edge of the pixel selection.

Arguments

`whatToDo, amount`
Acceptable values for *`whatToDo`* are `"expand"`, `"contract"`, `"border"`, and `"smooth"`. Any integer is acceptable for *`amount`*.

- Use `"expand"` to expand the pixel selection outward by the number of pixels that are specified by *`amount`*.
- Use `"contract"` to reduce the pixel selection inward by the number of pixels that are specified by *`amount`*.
- Use `"border"` to select a band of pixels the width of *`amount`* around the edge of the pixel selection.
- Use `"smooth"` to smooth out the edge of the pixel selection by *`amount`*.

Returns

Nothing.

dom.selectAll()

Availability

Fireworks 3

Description

Selects all the items in the current layer and frame. If single layer editing is enabled, all the items in the current layer are selected; otherwise, all elements on all layers are selected.

Arguments

None.

Returns

Nothing.

dom.selectAllOnLayer()

Availability

Fireworks MX

Description

Selects all the items on the given layer in the current frame. This function deselects objects on other layers. If the only element on the layer is a bitmap, Fireworks will enter paint mode on the bitmap.

Arguments

`layerIndex`

`layerIndex` is a long integer that identifies the layer on which to select the element.

Returns

Nothing.

dom.selectChildren()

Availability

Fireworks 3

Description

Selects the children, if any, of the selection. For example, if a group is selected, the selection changes from the group to the individual members of the group.

Arguments

None.

Returns

Nothing.

Related functions

`dom.selectParents()`

dom.selectFeather()

Availability

Fireworks 3

Description

If Fireworks is in bitmap mode and a pixel selection is active, this function feathers the selection by the specified number of pixels.

Arguments

`featherAmount`
`featherAmount` is an integer that specifies the number of pixels by which to feather the selection.

Returns

Nothing.

dom.selectInverse()

Availability

Fireworks 3

Description

If Fireworks is in bitmap mode and a pixel selection is active, this function inverts the pixel selection.

Arguments

None.

Returns

Nothing.

dom.selectNone()

Availability

Fireworks 3

Description

Deselects any selected items. If Fireworks is in image edit mode, has a pixel selection, and has a Selection tool selected, then this function deselects the pixels and exits image edit mode.

Arguments

None.

Returns

Nothing.

dom.selectParents()

Availability

Fireworks 3

Description

Selects the parents, if any, of the selection. That is, if all the members of a group are selected, the individual members are deselected, and the group is selected.

Arguments

None.

Returns

Nothing.

Related functions

`dom.selectChildren()`

dom.selectSimilar()

Availability

Fireworks 3

Description

If Fireworks is in bitmap mode and a pixel selection is active, this function selects all pixels in the current image that are within the specified tolerance of the average color in the current pixel selection.

Arguments

`tolerance`, *`edgemode`*, *`featherAmt`*, *`combinemode`*

- *`tolerance`* is an integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- Acceptable values for *`edgemode`* are `"hard edge"`, `"antialias"`, and `"feather"`.
- *`featherAmt`* is an integer that specifies the number of pixels to feather. This value is ignored if *`edgemode`* is not `"feather"`.
- *`combinemode`* specifies how to combine the new selection mask with the existing mask. Acceptable values are `"replace"`, `"add"`, `"subtract"`, and `"intersect"`.

Returns

Nothing.

Related functions

`dom.selectSimilarFromPoint()`

dom.selectSimilarFromPoint()

Availability

Fireworks 3

Description

Behavior is almost identical to `dom.selectSimilar()`, except that the new mask is calculated from the color at the specified location in the image, rather than from the average color in the selection.

Arguments

`where`, *`tolerance`*, *`edgemode`*, *`featherAmt`*, *`combinemode`*

- *`where`* is a point that specifies the *x,y* coordinates of the pixel whose color is used to calculate the new mask (see "Point" on page 6).
- *`tolerance`* is an integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- Acceptable values for *`edgemode`* are `"hard edge"`, `"antialias"`, and `"feather"`.
- *`featherAmt`* is an integer that specifies the number of pixels to feather. This value is ignored if *`edgemode`* is not `"feather"`.
- *`combinemode`* specifies how to combine the new selection mask with the existing mask. Acceptable values are `"replace"`, `"add"`, `"subtract"`, and `"intersect"`.

Returns

Nothing.

Related functions

`dom.selectSimilar()`

dom.setAllLayersDisclosure()

Availability

Fireworks 4

Description

Specifies whether all the elements in all layers appear in the Layers list.

Arguments

`bDisclosed`

If *`bDisclosed`* is `true`, all the elements on all layers appear in the Layers list. If `false`, only layer names appear on the list.

Returns

Nothing.

Related functions

`“dom.setLayerDisclosure()” on page 147`

dom.setAnimInstanceLoopCount()

Availability

Fireworks 3, deprecated in 4 in favor of “dom.setAnimInstanceNumFrames()” on page 127

Description

Sets the loop count of the selected instances of multiframe image symbols.

Arguments

`loopCount`

`loopCount` is an integer that corresponds to the loop count value that appears in the Objects panel when a multiframe image instance is selected.

Returns

Nothing.

dom.setAnimInstanceNumFrames()

Availability

Fireworks 4

Description

Sets the number of frames to animate the currently selected animation element.

Arguments

`numFrames`

`numFrames` is an integer that specifies the number of frames through which the symbol animates.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`

dom.setAnimInstanceOffsetDist()

Availability

Fireworks 4

Description

Sets the distance, in pixels, to animate the currently selected animation element.

Arguments

`offsetDistPt`

`offsetDistPt` is a point that specifies the distance the animation moves in pixels. For example, passing ({*x*:`100`, *y*:`25`}) animates the symbol to the right by 100 pixels and down by 25 pixels.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`

dom.setAnimInstanceRotationAmount()

Availability

Fireworks 4

Description

Sets the rotation amount, in degrees, to animate the currently selected animation element.

Arguments

`rotationAmount`

`rotationAmount` is a float value that specifies the degree of rotation to be applied to the animation symbol. For example, passing `720` specifies an animation that does two complete clockwise rotations. To rotate the animation counter-clockwise, pass a negative number.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`

dom.setAnimInstanceScaleAmount()

Availability

Fireworks 4

Description

Sets the scale amount to animate the currently selected animation instance.

Arguments

`scaleAmount`
`scaleAmount` is a positive float value that specifies the amount of scaling to be applied to the animation symbol. For example, passing `50` scales the symbol to 50 percent of its current size, and passing `200` scales it to twice its current size. To specify no scaling, pass `100`.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`

dom.setAnimInstanceStartEndOpacity()

Availability

Fireworks 4

Description

Sets the starting and ending opacity of the currently selected animation symbol.

Arguments

`startOpacity, endOpacity`
`startOpacity` and *`endOpacity`* are float values between 0 and 100 that specify the starting and ending opacity for the animation symbol.

Returns

Nothing.

Related functions

`dom.convertToAnimSymbol()`

dom.setAnimInstanceStartFrame()

Availability

Fireworks 3, deprecated in 4 in favor of placing the animation symbol on the frame in which it should start.

Description

Sets the start frame of the selected instances of multiframe image symbols.

Arguments

`startFrame`
`startFrame` is an integer that corresponds to the starting frame value that appears in the Objects panel when a multiframe image instance is selected.

Returns

Nothing.

dom.setBlendMode()

Availability

Fireworks 3

Description

Specifies the blend mode of the selection.

Arguments

`mode`

Acceptable values for *`mode`* are `"normal"`, `"multiply"`, `"screen"`, `"darken"`, `"lighten"`, `"difference"`, `"hue"`, `"saturation"`, `"color"`, `"luminosity"`, `"invert"`, `"tint"`, and `"erase"`.

Returns

Nothing.

dom.setBrush()

Availability

Fireworks 3

Description

Sets the selection to the specified brush.

Arguments

`brush`

`brush` is a `Brush` object (see "Brush" on page 21).

Returns

Nothing.

Related functions

```
dom.setBrushColor(), dom.setBrushName(), dom.setBrushNColorNTexture(),
  dom.setBrushPlacement()
```

dom.setBrushColor()

Availability

Fireworks 3

Description

Sets the brush color of the selection to the specified color.

Arguments

`color`

`color` is a color string (see "Color string" on page 5).

Returns

Nothing.

Related functions

```
dom.setBrushNColorNTexture()
```

dom.setBrushName()

Availability

Fireworks 3

Description

Renames a brush. Does not change the brush category.

Arguments

`category, currentName, newName`

- *`category`* is a string that specifies the category of the brush to be renamed.
- *`currentName`* is a string that specifies the current name of the brush.
- *`newName`* is a string that specifies the new name of the brush.

Returns

Nothing.

dom.setBrushNColorNTexture()

Availability

Fireworks 3

Description

Sets the selection to the specified brush, brush color, and brush texture.

Arguments

`brush, color, texture-name`

- *`brush`* is a `Brush` object (see "Brush" on page 21).
- *`color`* is a color string (see "Color string" on page 5).
- *`texture-name`* is the name of the texture to be applied.

Returns

Nothing.

Related functions

`dom.setBrushColor()`

dom.setBrushPlacement()

Availability

Fireworks 3

Description

Specifies the brush placement of the stroke on the selection.

Arguments

`placement`

Acceptable values for *`placement`* are `"inside"`, `"center"`, and `"outside"`.

Returns

Nothing.

dom.setButtonAutoSlice()

Availability

Fireworks 3

Description

If the user is editing a Button document, this function turns automatic slicing on or off.

Arguments

`bAutoSlice`
If *`bAutoSlice`* is `true`, automatic slicing is turned on. If *`bAutoSlice`* is `false`, it is turned off.

Returns

Nothing.

dom.setButtonIncludeDownState()

Availability

Fireworks 3

Description

If the user edits a Button document, this function specifies whether to include the "down" state in a button.

Arguments

`bIncludeDownState`
If *`bIncludeDownState`* is `true`, the "down" state is included in the button. If *`bIncludeDownState`* is `false`, it is not.

Returns

Nothing.

dom.setButtonIncludeOverWhileDownState()

Availability

Fireworks 3

Description

If the user edits a Button document, this function specifies whether to include the "over-while-down" state in a button.

Arguments

`bIncludeOverWhileDownState`
If *`bIncludeOverWhileDownState`* is `true`, the "over-while-down" state is included in the button. If *`bIncludeOverWhileDownState`* is `false`, it is not.

Returns

Nothing.

dom.setButtonShowDownOnLoad()

Availability

Fireworks 3

Description

If the user edits a Button document, this function specifies whether to show the "down-state-on-load" in a button.

Arguments

bShowDownOnLoad
If *bShowDownOnLoad* is `true`, the down-state-on-load is shown in the button. If *bShowDownOnLoad* is `false`, it is not.

Returns

Nothing.

dom.setButtonOptions()

Availability

Fireworks 3

Description

Sets the Button Export options. If the user edits a button, it sets options for the button being edited; if the user edits a normal document, it sets options for all the selected buttons.

Arguments

exportOptions, *URLString*, *altTagString*, *targetTagString*, *sliceName*, *statusMessage*

- *exportOptions* is an `ExportOptions` object (see "ExportOptions" on page 33).
- *URLString* is a string that specifies the URL for the button(s).
- *altTagString* and *targetTagString* specify the text for the button alt tag and target tag.
- *sliceName* is a string that specifies the name to be assigned to the slice that is associated with the button. If it is `null`, the slice is set to be named automatically.
- *statusMessage* is a string that specifies a status message to appear in the browser status line. If an empty string or `null` is passed, no status message appears.

Returns

Nothing.

dom.setDefaultBrushAndFillColors()

Availability

Fireworks 3

Description

Resets the document's brush and fill color to the default.

Arguments

None.

Returns

Nothing.

dom.setDefaultFillVector()

Availability

Fireworks 3

Description

Sets the fill-vector on the selection to the default.

Arguments

None.

Returns

Nothing.

dom.setDocumentCanvasColor()

Availability

Fireworks 3

Description

Sets the canvas color of the document to the specified color.

Arguments

`color`

`color` is a color string (see "Color string" on page 5).

Returns

Nothing.

Example

The following command sets the canvas color to blue.

```
fw.getDocumentDOM().setDocumentCanvasColor("#0000ff");
```

dom.setDocumentCanvasSize()

Availability

Fireworks 3

Description

Sets the document's canvas size to the specified rectangle.

Arguments

boundingRectangle

boundingRectangle is a rectangle that specifies the new canvas size for the document, in pixels (see "Rectangle" on page 6). Any items outside the specified rectangle are removed.

Returns

Nothing.

Example

The following command sets the canvas to a size of 200 by 200 pixels.

```
fw.getDocumentDOM().setDocumentCanvasSize({left:150, top:150, right:350,
  bottom:350});
```

dom.setDocumentCanvasSizeToDocumentExtents()

Availability

Fireworks 3

Description

Calculates the size of all the items in the document and resizes the document canvas to that size. This action is the same behavior as Modify > Trim Canvas.

Arguments

bGrowCanvas

If *bGrowCanvas* is `true`, the canvas can expand or shrink in size. If *bGrowCanvas* is `false`, it only shrinks.

Returns

Nothing.

Example

The following command resizes the canvas to include all the items in the document, enlarging the canvas if necessary.

```
fw.getDocumentDOM().setDocumentCanvasSizeToDocumentExtents(true);
```

Related functions

`dom.setDocumentCanvasSizeToSelection()`

dom.setDocumentCanvasSizeToSelection()

Availability

Fireworks 3

Description

Calculates the size of all the items in the selection and resizes the document canvas to that size.

Arguments

None.

Returns

Nothing.

Related functions

`dom.setDocumentCanvasSizeToDocumentExtents()`

dom.setDocumentImageSize()

Availability

Fireworks 3

Description

Scales the document to fit in the specified rectangle at the specified resolution.

Arguments

`boundingRectangle, resolution`

- *`boundingRectangle`* is a rectangle that specifies the size to which the document should be scaled (see “Rectangle” on page 6).
- *`resolution`* specifies the resolution for the scaled document (see “Resolution” on page 6).

Returns

Nothing.

dom.setDocumentResolution()

Availability

Fireworks 3

Description

Sets the resolution of the document.

Arguments

`resolution`

`resolution` specifies the resolution for the document (see “Resolution” on page 6).

Returns

Nothing.

dom.setEffectName()

Availability

Fireworks MX

Description

Sets the name for the current effect.

Arguments

`category`, *`oldName`*, *`newName`*

- *`category`* is a string that defines the name of the category of the effect.
- *`oldName`* is the existing name of the effect.
- *`newName`* is the new name to give to the effect.

Returns

Nothing.

dom.setElementMaskMode()

Availability

Fireworks 4

Description

Sets the rendering mode on the selected element's element mask. Only one element can be selected when calling this function. If selecting more than one element (or none) at the time this function is called, Fireworks throws an exception. Fireworks also returns an error if the selected element has no element mask.

Arguments

`mode`

Acceptable values for *`mode`* are `"mask to image"` and `"mask to path"`.

Returns

Nothing.

dom.setElementMaskShowAttrs()

Availability

Fireworks 4

Description

Specifies whether the currently selected vector mask shows the fill and stroke.

Arguments

`bShow`

If *`bShow`* is `true`, the vector mask fill and stroke are visible. If `false`, they are hidden.

Returns

Nothing.

dom.setElementName()

Availability

Fireworks 3

Description

Sets the name of the selected element(s).

Arguments

`name`

`name` is a string that specifies the name to be assigned to the selected element(s). To specify that no name should be assigned or that an existing name should be removed, pass `null`.

Returns

Nothing.

Related functions

`dom.findNamedElements()`

dom.setElementVisible()

Availability

Fireworks 4

Description

Shows or hides the specified element(s).

Arguments

`frameIndex, layerIndex, elementIndex, bShow`

- *`frameIndex`* is a zero-based integer that specifies the frame that contains the element(s) to be shown or hidden. To specify the current frame, pass `-1`.
- *`layerIndex`* is a zero-based integer that specifies the layer that contains the element(s) to be shown or hidden. To specify the current layer, pass `-1`.
- *`elementIndex`* is a zero-based integer that specifies the element(s) to show or hide, where 0 represents the topmost element in the specified layer. To show or hide all the elements in the specified layer, pass `-1`.
- If *`bShow`* is `true`, the element(s) are visible. If *`bShow`* is `false`, they are hidden.

Returns

Nothing.

Example

The following command hides all the elements in the current frame and layer.

```
fw.getDocumentDOM().setElementVisible(-1, -1, -1, false)
```

Related functions

`dom.setElementVisibleByName()`

dom.setElementVisibleByName()

Availability

Fireworks 4

Description

Shows or hides all the elements with the specified name. If no element has the specified name, an exception is thrown. If the elements are hidden because they are on a hidden layer or frame, for example, this function does not show them.

Arguments

`name`, *`bShow`*

- *`name`* is a string that specifies the name of the element(s) to be shown or hidden. If more than one element has the same name, this function shows or hides all of them.
- If *`bShow`* is `true`, the elements are visible. If *`bShow`* is `false`, they are hidden.

Returns

An array of the elements(s) for which visibility was set.

Related functions

`dom.findNamedElements()`, `dom.setElementName()`, `dom.setElementVisible()`

dom.setExportOptions()

Availability

Fireworks 3

Description

Sets the document Export Options.

Arguments

`exportOptions`
`exportOptions` is an `ExportOptions` object (see "ExportOptions" on page 33).

Returns

Nothing.

dom.setExportSettings()

Availability

Fireworks 3

Description

Sets the document Export Settings.

Arguments

`exportSettings`
`exportSettings` is an `ExportSettings` object (see "ExportSettings" on page 36).

Returns

Nothing.

dom.setFill()

Availability

Fireworks 3

Description

Sets the selection to the specified fill.

Arguments

`fill`

`fill` is a `Fill` object (see “Fill” on page 38).

Returns

Nothing.

dom.setFillColor()

Availability

Fireworks 3

Description

Changes the fill color of the selection to the specified color.

Arguments

`color`

`color` is a color string (see “Color string” on page 5).

Returns

Nothing.

dom.setFillEdgeMode()

Availability

Fireworks 3

Description

Sets the edge type for selected items with fills.

Arguments

`edgemode, featherAmt`

- Acceptable values for `edgemode` are `"hard edge"`, `"antialias"`, and `"feather"`.
- `featherAmt` is an integer that specifies the number of pixels to feather. This value is ignored if `edgemode` is not `"feather"`.

Returns

Nothing.

dom.setFillNColor()

Availability

Fireworks MX

Description

Sets the selection to the specified fill and fill color.

Arguments

`fill, color`

- *`fill`* is a `Fill` object (see “Fill” on page 38).
- *`color`* is a color string (see “Color string” on page 5).

Returns

Nothing.

dom.setFillNColorNTexture()

Availability

Fireworks 3

Description

Sets the selection to the specified fill, fill color, and fill texture.

Arguments

`fill, color, texture-name`

- *`fill`* is a `Fill` object (see “Fill” on page 38).
- *`color`* is a color string (see “Color string” on page 5).
- *`texture-name`* is the name of the texture to be applied.

Returns

Nothing.

Example

The following command sets the selected items to a linear fill with a feather edge and no texture.

```
fw.getDocumentDOM().setFillNColorNTexture({ category:"fc_Linear",
  ditherColors:[ "#000000", "#000000" ], edgeType:"antialiased", feather:10,
  gradient:{ name:"cn_WhiteBlack", nodes:[ { color:"#ffffff", position:0 }, {
  color:"#000000", position:1 } ] }, name:"fn_Normal", pattern:null,
  shape:"linear", stampingMode:"blend opaque", textureBlend:0,
  webDitherTransparent:false }, "#666666", "Grain");
```

dom.setFillPlacement()

Availability

Fireworks 3

Description

Sets the fill placement for selected items with fills.

Arguments

`placement`

Acceptable values for `placement` are `"top"` and `"bottom"`.

Returns

Nothing.

dom.setFillVector()

Availability

Fireworks 3

Description

Sets the fill vectors of the selection to the specified absolute values.

Arguments

`p1, p2, p3`

`p1`, *`p2`*, and *`p3`* are points that specify the *x,y* coordinates of the three points to be used in calculating the fill vector (see “Point” on page 6).

Returns

Nothing.

dom.setFillVectorStart()

Availability

Fireworks 3

Description

Modifies the fill vectors of the selection by moving the fill start to the specified point and then moving the two fill end handles to the same relative position.

Arguments

`p1`

`p1` is a point that specifies the *x,y* coordinates of the fill start and relative end handle placement to be used (see “Point” on page 6).

Returns

Nothing.

dom.setGradientName()

Availability

Fireworks 3

Description

Renames a gradient.

Arguments

`currentName, newName`

- `currentName` is a string that specifies the current name of the gradient.
- `newName` is a string that specifies the new name of the gradient.

Returns

Nothing.

dom.setGridOrigin()

Availability

Fireworks 3

Description

Sets the grid origin for the document.

Arguments

`gridOrigin`

`gridOrigin` is a point that specifies the *x,y* coordinates that are used for the document's grid origin (see "Point" on page 6).

Returns

Nothing.

dom.setGridSize()

Availability

Fireworks 3

Description

Sets the grid size for the document.

Arguments

`gridSize`

`gridSize` is a point that specifies the *x,y* coordinates that are used for the document's grid size (see "Point" on page 6).

Returns

Nothing.

dom.setGridColor()

Availability

Fireworks 3

Description

Sets the color used to display the grid.

Arguments

`gridColor`

`gridColor` is a color string (see "Color string" on page 5).

Returns

Nothing.

dom.setGroupType()

Availability

Fireworks 3, argument deprecated in 4

Description

Changes the group type for the currently selected groups.

Arguments

`{type}`

`type` is an optional string that specifies how to group the items. Acceptable values are `"normal"`, `"mask to image"`, and `"mask to path"`. If the argument is omitted, `"normal"` is assumed. (`"mask to image"` and `"mask to path"` are deprecated in 4.)

Returns

Nothing.

dom.setGuideColor()

Availability

Fireworks 3

Description

Sets the color that is used to display normal (nonslice) guides. To set the color of slice guides, use "dom.setSliceGuideColor()" on page 157.

Arguments

`guideColor`

`guideColor` is a color string (see "Color string" on page 5).

Returns

Nothing.

dom.setHotspotAltTag()

Availability

Fireworks 3

Description

Sets the alt tag text to the specified value for the hotspots and slices in the selection.

Arguments

`whatToSet, altTagString`

- Acceptable values for *`whatToSet`* are `"hotspots"`, `"slices"`, and `"hotspots and slices"`.
- *`altTagString`* is a string that specifies the text to be used for the alt tag.

Returns

Nothing.

Example

The following command sets the text attributes of the alt tag of the selected slices to `"This is my alt tag"`.

```
fw.getDocumentDOM().setHotspotAltTag("slices","This is my alt tag");
```

dom.setHotspotColor()

Availability

Fireworks 3

Description

Sets the color to the specified value for the hotspots and slices in the selection.

Arguments

`whatToSet, color`

- Acceptable values for *`whatToSet`* are `"hotspots"`, `"slices"`, and `"hotspots and slices"`.
- *`color`* is a color string (see "Color string" on page 5).

Returns

Nothing.

Example

The following command sets the color of the selected hotspots to the specified value, which, in this case, is red.

```
fw.getDocumentDOM().setHotspotColor("hotspots", "#ff0000");
```

dom.setHotspotRectangle()

Availability

Fireworks 3

Description

If the selection is a single hotspot or slice, this function moves or copies it to the specified location and size.

Arguments

`boundingRectangle`, *`bMakeCopy`*

- *`boundingRectangle`* is a rectangle that specifies the size of the new hotspot or slice (see "Rectangle" on page 6).
- *`bMakeCopy`* is a Boolean value; if it is `true`, the selection is copied and resized instead of moved and resized.

Returns

Nothing.

dom.setHotspotShape()

Availability

Fireworks 3

Description

Sets the shape to the specified value for the hotspots and slices in the selection.

Arguments

`whatToSet`, *`shape`*

- *`whatToSet`* can be `"hotspots"`, `"slices"`, or `"hotspots and slices"`.
- *`shape`* can be `"rectangle"`, `"oval"`, or `"polyline"`.

Returns

Nothing.

dom.setHotspotTarget()

Availability

Fireworks 3

Description

Sets the target tag text to the specified value for the hotspots and slices in the selection.

Arguments

`whatToSet`, *`targetTagString`*

- *`whatToSet`* can be `"hotspots"`, `"slices"`, or `"hotspots and slices"`.
- *`targetTagString`* is a string that specifies the text to be used for the target tag.

Returns

Nothing.

Example

The following command sets the currently selected slices to link to the parent window.

```
fw.getDocumentDOM().setHotspotTarget("slices", "_parent");
```

dom.setHotspotText()

Availability

Fireworks 3

Description

Sets the hotspot text to the specified value for the hotspots and slices in the selection.

Arguments

`whatToSet, textString, urlToMatch, bUpdateAttributes`

- *`whatToSet`* can be `"hotspots"`, `"slices"`, or `"hotspots and slices"`.
- *`textString`* is a string that specifies the text to be used for the hotspot or slice.
- *`urlToMatch`* is a string that specifies a URL that is already assigned to one or more hotspots in the document. If this value is not `null`, the URLs of all hotspots or slices in the document that have *`urlToMatch`* as their URL are changed to *`textString`*. Note: The URLs of both selected and unselected hotspots or slices are changed.
- If *`bUpdateAttributes`* is `true`, changed hotspots inherit the color, target, and alt tag text that were most recently associated with the new text value. For example, suppose *`textString`* is `"http://www.mywebsite.com"`, and the last time `"http://www.mywebsite.com"` was used, it was used with a color of blue, a target of none, and an alt tag of "Link to My Home Page". If *`bUpdateAttributes`* is `true`, any hotspot or slice whose text is now being changed to `"http://www.mywebsite.com"` will also have a color of blue, a target of none, and an alt tag text of `"Link to My Home Page"`.

Returns

Nothing.

Example

The following command creates a slice and inserts the HTML text, "`I am HTML text`".

```
fw.getDocumentDOM().setHotspotText("Slice ","I am HTML text", null, true);
```

dom.setLayerDisclosure()

Availability

Fireworks 4

Description

Specifies whether the elements on a specified layer appear in the Layers list. Disclosure affects the layer, regardless of which frame appears.

Arguments

`layerIndex, bDisclosed`

- *`layerIndex`* is a zero-based index that specifies the layer that contains the elements to be displayed or hidden. To specify the current layer, pass `-1`.
- If *`bDisclosed`* is `true`, all elements on the specified layer are displayed in the Layers list. If *`bDisclosed`* is `false`, only the layer name appears on the list.

Returns

Nothing.

Related functions

`dom.setAllLayersDisclosure()`

dom.setLayerLocked()

Availability

Fireworks 3

Description

Locks or unlocks one or all the layers on the specified frame.

Arguments

`layerIndex, frameIndex, bLock, bAllLayers`

- *`layerIndex`* is a zero-based integer that specifies the layer to be locked or unlocked. To specify the current layer, pass `-1`. (To lock or unlock all the layers on a frame, use the *`bAllLayers`* argument.)
- *`frameIndex`* is a zero-based integer that specifies the frame that contains the layer that is to be locked or unlocked. To specify the current frame, pass `-1`.
- If *`bLock`* is `true`, the layer is locked. If *`bLock`* is `false`, it is unlocked.
- If *`bAllLayers`* is `true`, all the layers on the specified frame are locked or unlocked, and any value passed for *`layerIndex`* is ignored.

Returns

Nothing.

Example

The following command locks all the layers on the first frame.

```
fw.getDocumentDOM().setLayerLocked(1, 0, true, true);
```

dom.setLayerName()

Availability

Fireworks 3

Description

Renames the specified layer. Layers aren't required to have unique names, so no duplicate checking occurs.

Arguments

`layerIndex, layerName`

- *`layerIndex`* is a zero-based integer that specifies the layer to be renamed. To specify the current layer, pass `-1`.
- *`layerName`* is a string that specifies the new name for the layer.

Returns

Nothing.

dom.setLayerSharing()

Availability

Fireworks 3

Description

Changes the “shared” layer status of a layer.

Arguments

`layerIndex`, *`sharedStatus`*, *`bUnshareCopiesToAllFrames`*, *`bWarnUser`*

- *`layerIndex`* is a zero-based integer that specifies the layer to be shared or not shared. To specify the current layer, pass `-1`.
- *`sharedStatus`* can be `"shared"` or `"not shared"`.
- *`bUnshareCopiesToAllFrames`* is used only if *`sharedStatus`* is `"not shared"` and the document has multiple frames. If these conditions are met and *`bUnshareCopiesToAllFrames`* is `true`, the items on the layer are duplicated to all the frames of the layer; if `false`, the items are placed only on the current frame.
- If *`bWarnUser`* is `true` and *`bUnshareCopiesToAllFrames`* is enabled, the user is asked to confirm that data on other frames can be overwritten. If *`bWarnUser`* is `false`, data on other frames of the layer is overwritten without warning.

Returns

Nothing.

Example

The following command sets the selected layer to “`shared`” and displays a warning that data loss is possible.

```
fw.getDocumentDOM().setLayerSharing(-1, "shared", false, true);
```

dom.setLayerVisible()

Availability

Fireworks 3

Description

Shows or hides a layer on the specified frame.

Arguments

`layerIndex`, *`frameIndex`*, *`bShow`*, *`bAllLayers`*

- *`layerIndex`* is a zero-based integer that specifies the layer that should be shown or hidden. To specify the current layer, pass `-1`. (To show or hide all the layers on a frame, use the *`bAllLayers`* argument.)
- *`frameIndex`* is a zero-based integer that specifies the frame that contains the layer to be shown or hidden. To specify the current frame, pass `-1`.
- If *`bShow`* is `true`, the layer is visible. If *`bShow`* is `false`, it is hidden.
- If *`bAllLayers`* is `true`, all the layers on the specified frame are shown or hidden, and any value that is passed for *`layerIndex`* is ignored.

Returns

Nothing.

dom.setMatteColor()

Availability

Fireworks 3

Description

Sets or removes the document's matte color that is used for exporting.

Arguments

`bUseMatteColor, matteColor`

- If *`bUseMatteColor`* is `true`, the document's matte color is set to the value that is specified by *`matteColor`*. If *`bUseMatteColor`* is `false`, any matte color is removed from the document, and the second argument is ignored.
- *`matteColor`* is a color string (see "Color string" on page 5).

Returns

Nothing.

Example

The following command sets the matte color to the specified value, which, in this case, is blue.

```
fw.getDocumentDOM().setMatteColor(true, "#0033ff");
```

dom.setPixelMask()

Availability

Fireworks 3, deprecated in 4 in favor of `dom.setSelectionMask()` (see "dom.setSelectionMask()" on page 153).

Description

If Fireworks is in bitmap mode, this function sets the pixel-selection mask of the current image to the specified mask.

Arguments

`mask, howToCombineMasks`

- *`mask`* is a mask variable that specifies the mask to be applied (see "Mask" on page 6). If ***mask*** is `null`, any existing pixel-selection mask is removed.
- If there was previously a mask and the new mask is also not `null`, then *`howToCombineMasks`* specifies how the two masks should be combined. Acceptable values for *`howToCombineMasks`* are `"replace"`, `"add"`, `"subtract"`, and `"intersect"`.

Returns

Nothing.

dom.setOnionSkinning()

Availability

Fireworks 3

Description

- Sets the onion-skin display options for the document.

Arguments

`before, after`

- The arguments are integers that specify the number of frames to display before and after the current one.
- To disable onion skinning, pass zero for both arguments.
- To enable onion skinning for all frames, pass `0` for the first argument and a large number for the second argument (for example, `99,999`).

Returns

Nothing.

Example

The following command turns on onion skinning two frames before the selected frame and zero frames after it.

```
fw.getDocumentDOM().setOnionSkinning(2, 0);
```

dom.setOpacity()

Availability

Fireworks 3

Description

Sets the opacity of the selection to the specified value.

Arguments

`opacity`

`opacity` is a float variable between 0 and 100, inclusive.

Returns

Nothing.

Example

The following command sets the selected item to an opacity of 55 percent.

```
fw.getDocumentDOM().setOpacity(55);
```

dom.setQuadrangle()

Availability

Fireworks 3

Description

Transforms the selection within a specified bounding quadrangle. The effect is the same as performing a Transform operation within Fireworks, and then replaying the Transform step from the History panel while other items are selected.

Arguments

pTopLeft, *pTopRight*, *pBottomRight*, *pBottomLeft*, *options*

- The first four arguments are points that specify the *x,y* coordinates of the top left, top right, bottom right, and bottom left points of the bounding rectangle (see "Point" on page 6).
- Acceptable values for *options* are `"transformAttributes"`, `"autoTrimImages"`, and `"autoTrimImages transformAttributes"`.

Returns

Nothing.

Example

The following command transforms the selection as specified.

```
fw.getDocumentDOM().setQuadrangle({x:-0.300884962, y:0.207964599}, {x:1,
  y:0.207964599}, {x:1, y:0.792035401}, {x:-0.300884962, y:0.792035401},
  "autoTrimImages transformAttributes");
```

dom.setRectRoundness()

Availability

Fireworks 4

Description

Modifies the corner roundness of all the selected rectangle primitives.

Arguments

roundness

roundness is a float value between 0 and 1 that specifies the roundness to use for the corners (0 is no roundness, 1 is 100 percent roundness).

Returns

Nothing.

Related functions

`dom.addNewRectanglePrimitive()`, `dom.setRectSides()`

dom.setRectSides()

Availability

Fireworks 4

Description

Modifies the untransformed sides of all selected rectangle primitives.

Arguments

newSides

newSides is a rectangle that specifies the new untransformed sides of the rectangle primitive (see "Rectangle" on page 6). Rectangle primitives remember their transformations, so the user sees the transformed result of *newSides* in the document.

Returns

Nothing.

Related functions

`dom.setRectRoundness()`, `dom.addNewRectanglePrimitive()`

dom.setSelectionBounds()

Availability

Fireworks 3

Description

Moves and resizes the selection in a single operation.

Arguments

`boundingRectangle`, *`opts`*

- *`boundingRectangle`* is a rectangle that specifies the new location and size of the selection (see "Rectangle" on page 6).
- Acceptable values for *`opts`* are `"transformAttributes"`, `"autoTrimImages"`, and `"autoTrimImages transformAttributes"`.

Returns

Nothing.

dom.setSelectionMask()

Availability

Fireworks 4

Description

If Fireworks is in bitmap mode, this function sets the pixel-selection mask of the current image to the specified mask.

Arguments

`mask`, *`howToCombineMasks`*

- *`mask`* specifies the mask to be applied (see "Mask" on page 6). If *`mask`* is `null`, an existing pixel-selection mask is removed.
- If there was previously a mask and *`mask`* is not `null`, *`howToCombineMasks`* specifies how the two masks should be combined. Acceptable values are `"replace", "add", "subtract",` and `"intersect".`

Returns

Nothing.

dom.setShowEdges()

Availability

Fireworks 3

Description

Specifies whether the Show Edges option is on or off.

Arguments

`bShowEdges`
If *`bShowEdges`* is `true`, the Show Edges option is turned on. If *`bShowEdges`* is `false`, the option is turned off.

Returns

Nothing.

dom.setShowGammaPreview()

Availability

Fireworks 3

Description

Specifies whether the Preview Gamma option is on or off.

Arguments

`bPreviewGamma`
If *`bPreviewGamma`* is `true`, the Preview Gamma option is turned on. If *`bPreviewGamma`* is `false`, the option is turned off.

Returns

Nothing.

dom.setShowGrid()

Availability

Fireworks 3

Description

Specifies whether the grid is visible.

Arguments

`bShow`
If *`bShow`* is `true`, the grid is visible. If *`bShow`* is `false`, it is not visible.

Returns

Nothing.

dom.setShowGuides()

Availability

Fireworks 3

Description

Specifies whether normal guides are visible.

Arguments

bShow

If *bShow* is `true`, the normal guides are visible. If *bShow* is `false`, they are not visible.

Returns

Nothing.

dom.setShowRulers()

Availability

Fireworks 3

Description

Specifies whether rulers are visible.

Arguments

bShow

If *bShow* is `true`, the rulers are visible. If *bShow* is `false`, they are not visible.

Returns

Nothing.

dom.setShowSliceGuides()

Availability

Fireworks 3

Description

Specifies whether slice guides are visible.

Arguments

bShow

If *bShow* is `true`, the slice guides are visible. If *bShow* is `false`, they are not visible.

Returns

Nothing.

dom.setShowSliceOverlay()

Availability

Fireworks 3

Description

Specifies whether the slice overlay is visible.

Arguments

`bShow`

If *`bShow`* is `true`, the slice overlay is visible. If *`bShow`* is `false`, it is not visible.

Returns

Nothing.

dom.setSliceAutonaming()

Availability

Fireworks 3

Description

If a single slice is selected, this function turns automatic naming on or off for the slice.

Arguments

`bAutoname`

If *`bAutoname`* is `true`, automatic naming is turned on for the slice. If *`bAutoname`* is `false`, it is turned off.

Returns

Nothing.

dom.setSliceExportOptions()

Availability

Fireworks 3

Description

Sets the Export Options for the selected slices.

Arguments

`exportOptions`

`exportOptions` is an `ExportOptions` object (see “ExportOptions” on page 33).

Returns

Nothing.

dom.setSliceFilename()

Availability

Fireworks 3

Description

If a single slice is selected, this function turns off automatic naming for the slice and sets its filename to the specified URL.

Arguments

`fileURL`

`fileURL` is a string, which is expressed as a file://URL, that specifies the name to be given to the slice.

Returns

Nothing.

dom.setSliceGuideColor()

Availability

Fireworks 3

Description

Sets the color that is used to display slice guides. To set the color of normal guides, use `dom.setGuideColor()`.

Arguments

`color`

`color` is a color string (see “Color string” on page 5).

Returns

Nothing.

dom.setSliceHtml()

Availability

Fireworks 3

Description

If a single slice is selected, this function sets the slice’s HTML text.

Arguments

`htmlText`

`htmlText` is a string that specifies the HTML text for the slice.

Returns

Nothing.

dom.setSliceIsHtml()

Availability

Fireworks 3

Description

Sets the selected slices as HTML or Image.

Arguments

`bHtml`

If *`bHtml`* is `true`, it sets the slices as HTML. If *`bHtml`* is `false`, it sets the slices as Image.

Returns

Nothing.

dom.setSnapToGrid()

Availability

Fireworks 3

Description

Specifies whether tools snap to grid.

Arguments

`bSnap`

If *`bSnap`* is `true`, the tools snap to grid. If *`bSnap`* is `false`, they do not.

Returns

Nothing.

dom.setSnapToGuides()

Availability

Fireworks 3

Description

Specifies whether tools snap to guides.

Arguments

`bSnap`

If *`bSnap`* is `true`, the tools snap to all guides. If *`bSnap`* is `false`, they do not.

Returns

Nothing.

dom.setSymbolProperties()

Availability

Fireworks 3

Description

Sets the name and symbol type of the specified symbol.

Arguments

`currentName, symbolType, newName`

- *`currentName`* specifies the current name of the symbol in the library. If more than one master exists with a name of *`currentName`*, only the first master is changed. If `null` is passed in for *`currentName`*, the name property is set for all selected symbols in the library (not the document).
- Acceptable values for *`symbolType`* are `"graphic"`, `"button"`, and `"animation"`.
- *`newName`* specifies the new name for the symbol.

Returns

Nothing.

dom.setTextAlignment()

Availability

Fireworks 3

Description

Sets the alignment of the selected text items to the specified setting.

Arguments

`alignment`

Acceptable values for *`alignment`* are `"left"`, `"center"`, `"right"`, `"justify"`, `"stretch"`, `"vertical left"`, `"vertical center"`, `"vertical right"`, `"vertical justify"`, and `"vertical stretch"`.

Returns

Nothing.

dom.setTextAntiAliasing()

Availability

Fireworks 3

Description

Sets the anti-aliasing level for the selected blocks of text.

Note: To turn anti-aliasing on or off, call "dom.enableTextAntiAliasing()" on page 89.

Arguments

`level`

Acceptable values for *`level`* are `"crisp"`, `"smooth"`, and `"strong"`.

Returns

Nothing.

dom.setTextAutoKern()

Availability

Fireworks 3

Description

Specifies whether automatic kerning is on or off for the selected text items.

Arguments

`bKern`

If *`bKern`* is `true`, automatic kerning is on for the selected text items. If *`bKern`* is `false`, it is off.

Returns

Nothing.

dom.setTextCharSpacing()

Availability

Fireworks MX

Description

Sets the kerning for text.

Arguments

`charSpace`

`charSpace` is a floating-point percentage of the default spacing to add to (positive values) or remove from (negative values) the space between two adjacent characters. To increase the spacing by 15 percent, for example, pass `0.15`.

Returns

Nothing.

dom.setTextFlow()

Availability

Fireworks 3

Description

Sets the horizontal flow direction of the selected text items.

Arguments

`flowDirection`

Acceptable arguments for *`flowDirection`* are `"left to right"` and `"right to left"`.

Returns

Nothing.

dom.setTextHorizontalScale()

Availability

Fireworks MX

Description

Sets the horizontal scaling of text. For vertical text mode, this function stretches or compresses the height of the characters.

Arguments

`horizScale`

`horizScale` is a floating-point number that describes how much to scale the text characters horizontally. 1.0 is normal. Values greater than 1.0 make the characters wider, and values less than 1.0 make the characters narrower.

Returns

Nothing.

dom.setTextLeading()

Availability

Fireworks MX

Description

Sets the leading amount and leading mode for text. For vertical text mode, the leading is the space between two adjacent columns of text.

Arguments

`leadingValue`, *`leadingMode`*

- *`leadingValue`* is a floating-point number that determines the spacing between two lines of text. The exact meaning of `leadingValue` depends on `leadingMode`.
- *`leadingMode`* can be either `"exact"` or `"percentage"`. `"exact"` means the `leadingValue` is the number of pixels between two lines of text. `"percentage"` means the `leadingValue` is a percentage of the default leading amount; 1.0 is normal, 0.5 is close together, and 2.0 is double-spaced.

Returns

Nothing.

dom.setTextOnPathMode()

Availability

Fireworks 3

Description

Sets the mode of the selected text-on-a-path items to the specified value.

Arguments

`mode`
Acceptable values for *`mode`* are `"rotate"`, `"vertical"`, `"skew vertical"`, and `"skew horizontal"`.

Returns

Nothing.

dom.setTextOnPathOffset()

Availability

Fireworks 3

Description

Sets the offset for the selected text-on-a-path items to the specified distance.

Arguments

`offset`
`offset` is a float value that specifies the offset distance in pixels.

Returns

Nothing.

dom.setTextOrientation()

Availability

Fireworks 3

Description

Sets the horizontal/vertical text orientation of the selected text items.

Arguments

`orientation`
Acceptable values for *`orientation`* are `"horizontal left to right"`, `"vertical right to left"`, `"horizontal right to left"`, and `"vertical left to right"`.

Returns

Nothing.

dom.setTextParaIndent()

Availability

Fireworks MX

Description

Sets the paragraph indent for text. Paragraph indent is the amount to indent the first line of a paragraph in pixels.

Arguments

`paraIndent`

`paraIndent` is the number of pixels to indent the first line of a paragraph.

Returns

Nothing.

dom.setTextParaSpacingAfter()

Availability

Fireworks MX

Description

Sets the after-paragraph spacing for text; that is, the number of pixels to move down after a paragraph before starting the next paragraph. For vertical text mode, this function defines the distance to move vertically before or after starting a new paragraph.

Arguments

`paraSpaceAfter`

`paraSpaceAfter` is the number of pixels to place after a paragraph before starting the next paragraph.

Returns

Nothing.

dom.setTextParaSpacingBefore()

Availability

Fireworks MX

Description

Sets the before-paragraph spacing for text; that is the number of pixels to move down before starting a new paragraph. For vertical text mode, this function defines the distance to move vertically before or after starting a new paragraph.

Arguments

`paraSpaceBefore`

`paraSpaceBefore` is the number of pixels to move down before starting a new paragraph.

Returns

Nothing.

dom.setTextRuns()

Availability

Fireworks 3

Description

Replaces the text in the selected text blocks with the styled text that is described by the `TextRuns` object passed in the argument.

Arguments

`textRuns`
The argument is a `TextRuns` object (see “TextRuns” on page 44).

Returns

Nothing.

dom.setTransformMode()

Availability

Fireworks 3

Description

Sets the transform mode for the selected text, instance items, or both.

Arguments

`mode`
Acceptable values for *`mode`* are `"paths"` and `"pixels"`.

Returns

Nothing.

dom.setTextRectangle()

Availability

Fireworks 3

Description

Changes the bounding rectangle for the selected text item to the specified size. This function causes the text to reflow inside the new rectangle; the text item is not scaled or transformed. Text that does not fit into the new rectangle does not show.

Arguments

`boundingRectangle`
`boundingRectangle` is a rectangle that specifies the new size within which the text item should flow (see “Rectangle” on page 6).

Returns

Nothing.

dom.setTextRectangleAuto()

Availability

Fireworks 3

Description

Recalculates the bounding rectangle for the selected text item, setting the rectangle to the smallest box that encloses the text.

Arguments

None.

Returns

Nothing.

Related functions

`dom.setTextRectangleAutoFromPoint()`

dom.setTextRectangleAutoFromPoint()

Availability

Fireworks 3

Description

Performs the same function as `dom.setTextRectangleAuto()`, but lets you pass a point to specify where the rectangle should be located.

Arguments

`anchorPoint`

`anchorPoint` is a point that specifies the *x,y* coordinates of the location at which the text box should be anchored (see “Point” on page 6). How the point is used depends on the left-to-right and up-to-down orientation of the text flow in the text block.

- Left-justified horizontal text is placed with its top and left edges at *`anchorPoint`*, and the text expands to the right.
- Centered horizontal text is centered horizontally around *`anchorPoint`* and expands equally to the left and right.
- Centered vertical text is centered vertically around *`anchorPoint`* and expands equally up and down.

Returns

Nothing.

Related functions

`dom.setTextRectangleAuto()`

dom.showAllHidden()

Availability

Fireworks 3

Description

Shows all the items that were hidden by using `dom.hideSelection()`.

Arguments

None.

Returns

Nothing.

dom.splitPaths()

Availability
Fireworks 3

Description
Splits the selected paths. Compound paths are split into separate contours.

Arguments
None.

Returns
Nothing.

dom.swapBrushAndFillColors()

Availability
Fireworks 3

Description
Swaps the current brush color and current fill color. This function has no effect on any selected items.

Arguments
None.

Returns
Nothing.

dom.transformSelection()

Availability
Fireworks 3, enhanced in 4

Description
Transforms the selection using the specified three-by-three matrix.

Arguments
`matrix, options`

- *`matrix`* is a three-by-three transformation matrix (see "Matrix" on page 6).
- Acceptable values for *`options`*, some of which were added in Fireworks 4, are `""`, `"transformAttributes"`, `"autoTrimImages"`, `"autoTrimImages transformAttributes"`, `"rememberQuad"`, `"transformAttributes rememberQuad"`, `"autoTrimImages rememberQuad"`, and `"autoTrimImages transformAttributes rememberQuad"`.

Returns
Nothing.

dom.tween()

Availability

Fireworks 3

Description

Tweens between the two selected instances.

Arguments

`numSteps`, *`bDistribute`*

- *`numSteps`* is an integer that specifies how many new instances are generated.
- If *`bDistribute`* is `true`, the new instances are distributed to frames.

Returns

Nothing.

dom.undo()

Availability

Fireworks 3

Description

Undoes the most recent step performed, as long as that step is actually "undoable." Most (but not all) JavaScript functions create an "undoable" action to be executed.

Arguments

None.

Returns

Nothing.

dom.updateSymbol()

Availability

Fireworks 3

Description

Updates the specified linked symbol.

Arguments

`name`

`name` specifies the name of the symbol in the library. If more than one symbol exists with a name of *`name`*, then only the first symbol with that name is updated. If `null` is passed in for *`name`*, then all the selected linked symbols in the library (not the document) are updated.

Returns

Nothing.

dom.ungroup()

Availability

Fireworks 3

Description

Ungroups any grouped items in the selection. To group items, use `dom.group()`.

Arguments

None.

Returns

Nothing.

Fireworks functions

In Fireworks MX, `fw` is synonymous with `fireworks`. All methods of the `fireworks` object can be referred to as `fireworks.`*`functionName()`* or as `fw.`*`functionName()`*.

fw.browseDocument()

Availability

Fireworks 3

Description

Opens the user's primary browser and displays the specified URL.

Arguments

`URL`

`URL` is the URL of the page appear in the browser. Any legal URL (including http://, ftp://, and so on) can be passed. Fireworks does not check this argument for syntax; if you pass an illegal value, the browser does not open the URL.

Returns

Nothing.

fw.browseForFileURL()

Availability

Fireworks 3

Description

Displays an Open or Save dialog box for the user.

Arguments

`browseType, title, previewArea`

- Acceptable values for *`browseType`* are `"open"`, `"select"`, and `"save"`. The first two values display an Open dialog box; each is acceptable for compatibility with Dreamweaver. The third value displays a Save dialog box.
- *`title`* and *`previewArea`* are ignored by Fireworks but are accepted for compatibility with Dreamweaver.

Returns

`The file URL selected by the user`, or `null` if the dialog box was canceled.

fw.browseForFolderURL()

Availability

Fireworks 3

Description

Displays a dialog box that lets a user select a particular directory.

Arguments

`{title}, {startFolder}`

- *`title`* is an optional string that specifies a title for the dialog box that appears. If it is omitted or `null`, a default title appears.
- *`startFolder`* is an optional string that serves as the root directory for the dialog box that appears. If it is omitted or `null`, the browse dialog box displays an unspecified directory, depending on your system configuration. Generally, it is the last directory used.

fw.browseHelp()

Availability

Fireworks MX

Description

Opens the specified help topic in the help viewer.

Arguments

`helpID`

`helpID` is the index number of the help topic to view.

Returns

Nothing.

fw.checkFwJsVersion()

Availability

Fireworks 3

Description

Checks the JavaScript API for incompatibilities.

Arguments

`version`

`version` is an integer that is reserved for future use; only a value of 0 is supported at this time. To use this function, put a call to *`fw.checkFwJsVersion(0)`* in your script.

Returns

Nothing.

fw.chooseBrowser()

Availability

Fireworks MX

Description

Displays a dialog box that lets the user select a primary or secondary browser.

Arguments

`primaryBrowser`

`primaryBrowser` is a Boolean value that indicates which browser to select. If `primaryBrowser` is `true`, Fireworks prompts the user to set the primary browser; if the argument is `false`, Fireworks prompts the user to set the secondary browser.

Returns

Nothing.

fw.chooseScriptTargetDialog()

Availability

Fireworks 4

Description

Displays a dialog box that lets the user choose the target document(s) for an operation. The dialog box lets the user specify the files currently open, the files in the project list, or files that are explicitly selected.

Arguments

`formatlist`
`formatlist` is similar to `fw.locateDocDialog()`, except that `formatlist` is required, and you cannot specify a maximum number of documents (see "fw.locateDocDialog()" on page 183).

Returns

An array of file://URLs, or `null` if the dialog box is canceled.

fw.closeDocument()

Availability

Fireworks 3

Description

Closes the specified document.

Arguments

`document, {bPromptToSaveChanges}`

- `document` is a `Document` object that specifies the document to close (see "Document" on page 9).
- If `bPromptToSaveChanges` is `true` or omitted, and the document has changed since the last time it was saved, the user is prompted to save changes to the document. If `bPromptToSaveChanges` is `false`, the user is not prompted and any changes to the document are discarded.

Returns

Nothing.

fw.createDocument()

Availability

Fireworks 3

Description

Opens a new document and selects it. Values for size, resolution, and color are the same as the current defaults. To specify values other than the defaults, use `fw.createFireworksDocument()`.

Arguments

None.

Returns

The `Document` object for the newly created document (see "Document" on page 9).

fw.createFireworksDocument()

Availability

Fireworks 3

Description

Opens a new document and selects it. Values for size, resolution, and color are explicitly specified. To open a new document with the default values, use `fw.createDocument()`.

Arguments

`size, res, backgroundColor`

- *`size`* is a point whose *x* value specifies the document's width and whose *y* value specifies the document's height. Both values are pixels.
- *`res`* specifies the resolution for the scaled document (see "Resolution" on page 6).
- *`backgroundColor`* is a color string (see "Color string" on page 5).

Returns

The `Document` object for the newly created document (see "Document" on page 9).

Example

The following command creates a new document that is 500 by 500 pixels in size, with a resolution of 72 dpi, and a solid white background color.

```
fw.createFireworksDocument({x:500,y:500},{pixelsPerUnit:72,units:"inch"},
  "#ffffff");
```

fw.dismissBatchDialogWhenDone()

Availability

Fireworks 4

Description

Closes the Batch Progress dialog box automatically when the script finishes. This function has no effect if the Batch Progress dialog box does not appear.

Note: This function is used mostly for backward compatibility with Fireworks 2.

Arguments

`autoClose`

`autoClose` is a Boolean value. If set to `true`, the Batch Progress dialog box closes automatically (without user intervention) when the script finishes.

Returns

Nothing.

fw.exportAndCopyHTMLCode()

Availability

Fireworks MX

Description

Displays the export dialog box, which is preconfigured to export HTML and images and to copy the HTML code to the Clipboard.

Arguments

`document`

`document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If `document` is `null`, the active document is exported.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportDirectorAsSlices()

Availability

Fireworks MX

Description

Exports the specified document to the specified file as Director images.

Arguments

`document, fileURL`

- `document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If `document` is `null`, the active document is exported.
- `fileURL` specifies the filename for the exported file. If `fileURL` is `null`, Fireworks displays the Export dialog box.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportDirectorAsLayers()

Availability

Fireworks MX

Description

Exports the specified document to the specified file as layers to be imported into Macromedia Director.

Arguments

`document, fileURL`

- `document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If `document` is `null`, the active document is exported.
- `fileURL` specifies the filename for the exported file. If `fileURL` is `null`, Fireworks displays the Export dialog box.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportDocumentAs()

Availability

Fireworks 3

Description

Exports the specified document to the specified file.

Arguments

`document, fileURL, exportOptions`

- `document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to be exported. If `document` is `null`, the active document is exported.
- `fileURL` is a string, which is expressed as a file://URL, that specifies the filename for the exported file. If `fileURL` is `null`, the Save As dialog box is displayed.
- `exportOptions` is an `ExportOptions` object (see "ExportOptions" on page 33). If `exportOptions` is `null`, the document's current export options are used. If the file format specified by `exportOptions` conflicts with the file format specified by `fileURL`, then the extension of `fileURL` is changed to match the format specified by `exportOptions`.

Returns

Nothing.

Related functions

`fw.exportHtmlAndImages()`

fw.exportIllustrator()

Availability

Fireworks MX

Description

Exports the specified document to the specified file in Adobe Illustrator format.

Arguments

`document, fileURL`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If *`document`* is `null`, the active document is exported.
- *`fileURL`* specifies the filename for the exported file. If *`fileURL`* is `null`, Fireworks displays the Export dialog box.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportPSD()

Availability

Fireworks MX

Description

Exports the specified document to the specified file in Adobe Photoshop format.

Arguments

`document, fileURL`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If *`document`* is `null`, the active document is exported.
- *`fileURL`* specifies the filename for the exported file. If *`fileURL`* is `null`, Fireworks displays the Export dialog box.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportSWF()

Availability

Fireworks MX

Description

Exports the specified document to the specified file in Macromedia Flash SWF format.

Arguments

`document, fileURL`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document to export. If `document` is `null`, the active document is exported.
- *`fileURL`* specifies the filename for the exported file. If *`fileURL`* is `null`, Fireworks displays the Export dialog box.

Returns

A Boolean value: `true` if successful; `false` otherwise.

fw.exportFrames()

Availability

Fireworks 4

Description

Exports a document's frames as individual images. The images are named based on the names in the Frames panel.

Arguments

`docObject, directoryURL`

- `docObject` is a `Document` object that specifies the document that contains the frames to export (see "Document" on page 9). To export frames from the current document, pass `null`.
- `directoryURL` is the directory where the images will be placed, which is expressed as a file:// URL.

Example

The following command exports the frames in the current document to the C:\images directory.

```
fw.exportFrames(null, "file:///C|/images");
```

fw.exportHtmlAndImages()

Availability

Fireworks 4

Description

Exports one image if the document contains no slice objects and multiple images if the document contains one or more slice objects. It also optionally exports HTML. The document is exported using the current export settings and export options.

Arguments

`doc, htmlUrl, imagesUrl`

- `doc` is a `Document` object that specifies the document to be exported (see "Document" on page 9). If `doc` is `null`, the active document is exported.
- `htmlUrl` is the filename for the exported HTML file, which is expressed as a file://URL. If `htmlUrl` is `null`, no HTML is generated.
- `imagesUrl` is the filename for the exported image(s), which is expressed as a file://URL, and might not be `null`. If a single image is generated, this function uses `imagesUrl` as the filename for the image. If multiple sliced images are exported, it uses `imagesURL` to generate automatically named images, and all images are placed in this directory.

Returns

Nothing.

Example

The following command exports the current document as HTML and as one or more images.

```
fw.exportHtmlAndImages(null, "file:///C|/mysite/nav.htm", "file:///C|/mysite/
  images/nav.gif");
```

Related functions

`fw.exportDocumentAs()`

fw.exportLayers()

Availability

Fireworks 4

Description

Exports a document's layers as individual images. The images are named based on the names in the Layers panel. The layers from the current frame are exported.

Arguments

`docObject, directoryURL`

- *`docObject`* is a `Document` object that specifies the document that contains the layers to export (see "Document" on page 9). To export layers from the current document, pass `null`.
- *`directoryURL`* is the directory in which the images will be placed, which is expressed as a file:/ /URL.

Example

The following command exports the layers in the third open document to the C:\images directory.

```
fw.exportLayers(fw.documents[2], "file:///C|/images");
```

fw.exportPSD()

Availability

Fireworks 4

Description

Exports a Fireworks document as a Photoshop document.

Arguments

`docObject, PSDDocumentURL`

- *`docObject`* is a `Document` object that specifies the document to export (see "Document" on page 9). To export the current document, pass `null`.
- *`PSDDocumentURL`* is the name of the Photoshop document to be created, which is expressed as a file://URL.

Example

The Photoshop writer is controlled by the values of several preferences. See the following example for allowed values. A well-behaved script should restore the original values after exporting the file.

```
var prevWarn = fw.getPref("PsdExport_Warn100"); // bool
fw.setPref("PsdExport_Warn100", false);  // don't warn.

var kObjToLayer = 1;
var kFlatten = 2;
var prevLayers = fw.getPref("PsdExport_Layers");
fw.setPref("PsdExport_Layers", kObjToLayer);  // flatten layers or not.

var kEffectEditable = 1;
var kEffectRender = 2;
var prevEffects = fw.getPref("PsdExport_Effects");
fw.setPref("PsdExport_Effects", kEffectEditable);

var kTextEditable = 1;
var kTextRender = 2;
var prevText = fw.getPref("PsdExport_Text");
fw.setPref("PsdExport_Text", kTextRender);

fw.exportPSD(null, "file:///C|/new folder/test.psd");

// Put the prefs back.
fw.setPref("PsdExport_Warn100", prevWarn);
fw.setPref("PsdExport_Layers", prevLayers);
fw.setPref("PsdExport_Effects", prevEffects);
fw.setPref("PsdExport_Text", prevText);
```

fw.exportSWF()

Availability

Fireworks 4

Description

Exports a Fireworks document as a Macromedia Flash document.

Arguments

`docObject`, *`FlashDocumentURL`*

- *`docObject`* is a `Document` object that specifies the document to be exported (see "Document" on page 9). To export the current document, pass `null`.
- *`FlashDocumentURL`* is the name of the Macromedia Flash document to be created, which is expressed as a file://URL.

Example

The Macromedia Flash writer is controlled by the values of several preferences. See the following example for allowed values. A well-behaved script should restore the original values after exporting the file.

```
var prevMaintainObjEditable = fw.getPref("SwfMaintainObjEditable");
fw.setPref("SwfMaintainObjEditable", true);
  // maintain non-text editability
  //at expense of appearance or not

var prevMaintainTextEditable = fw.getPref("SwfMaintainTextEditable");
fw.setPref("SwfMaintainTextEditable", false);
  // maintain text editability
  // at expense of appearance or not

var prevExportAllFrames = fw.getPref("SwfExportAllFrames");
fw.setPref("SwfExportAllFrames", true);
  // if true all frames are exported

var prevExportFromFrame = fw.getPref("SwfExportFromFrame");
fw.setPref("SwfExportFromFrame", 1);
  // from frame; only used ifSwfExportAllFrames is false
var prevExportToFrame = fw.getPref("SwfExportToFrame");
fw.setPref("SwfExportToFrame", 5);
  // from frame; only used if SwfExportAllFrames is false

var prevJpegQualit = fw.getPref("SwfJpegQuality");
fw.setPref("SwfJpegQuality", 85);  // JPEG quality

var prevFrameRate = fw.getPref("SwfFrameRate");
fw.setPref("SwfFrameRate", 5);  // frame rate

fw.exportSWF(null, "file:///C|/new folder/test.swf");

// Put the prefs back.
fw.setPref("SwfMaintainObjEditable", prevMaintainObjEditable);
fw.setPref("SwfMaintainTextEditable", prevMaintainTextEditable);
fw.setPref("SwfExportAllFrames", prevExportAllFrames);
fw.setPref("SwfExportFromFrame", prevExportFromFrame);
fw.setPref("SwfExportToFrame", prevExportToFrame);
fw.setPref("SwfJpegQuality", prevJpegQualit);
fw.setPref("SwfFrameRate", prevFrameRate);
```

fw.findApp()

Availability

Fireworks MX

Description

Attempts to find the path to the requested application. On the Macintosh, Fireworks looks for the application using a four-character signature code. On Windows, Fireworks looks in the Windows registry under
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths`.

Arguments

`macAppSignature or winExeRegistryName`

- *`macAppSignature`* is a Macintosh-specific string that identifies the signature of the application to find, such as `"MKBY"`.
- *`winExeRegistryName`* is a Windows-specific string that identifies the name of an executable to find in the Windows registry, such as `"Fireworks.exe"`.

Returns

A URL to the application. This URL can be passed as an argument to `fw.launchApp()` on page 182. If no such application can be found, the URL is empty.

fw.findNext()

Availability

Fireworks 3

Description

Finds the next instance of the current search string and selects that section of the document. To begin a search, use `fw.setUpFindReplace()`.

Arguments

None.

Returns

The number of items that are replaced if the search is completed, or `-1` if there are items in the document that remain to be searched.

fw.findOpenDocument()

Availability

Fireworks 3

Description

Determines whether the specified file is open in a Fireworks document window.

Arguments

`docname`
`docname` is a string that specifies the name of the document, which is expressed as a file://URL.

Returns

If the document is open, it returns the Document object; otherwise, it returns `null` (see "Document" on page 9).

fw.getDocumentDOM()

Availability

Fireworks 3

Description

Returns the `Document` object for the active document (see "Document" on page 9).

Arguments

`{`*`which-string`*`}`
`which-string` is an optional string that is included for compatibility with Dreamweaver. If specified here, it must be `"`*`document`*`"`.

Returns

The `Document` object for the active document, or `null` if no document is open.

fw.getDocumentPath()

Availability

Fireworks 3

Description

Gets the path and filename of the specified document.

Arguments

`document`

The *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document whose path and filename should be retrieved. If *`document`* is `null`, information about the active document is retrieved.

Returns

The file URL for the document if it was saved or an empty string if it has not been saved.

fw.getFloaterGroupings()

Availability

Fireworks 3

Description

Gets an array of arrays that indicates the tab-grouping of the panels (even hidden ones).

Arguments

None.

Returns

An array that looks like the following example:

```
[ [ "stroke", "fill", "effect" ], [ "layers", "frames", "object" ], [ "mixer",
  "options", "swatches", "info" ], [ "styles", "library" ], [ "find", "project
  log" ], [ "url" ], [ "optimize", "optimized colors" ], [ "behaviors" ], [
  "history" ] ]
```

fw.getFloaterPosition()

Availability

Fireworks 3

Description

Gets the screen position and size of the specified panel.

Arguments

`panelName`

Acceptable values for *`panelName`* are `"find"`, `"project log"`, `"object"`, `"info"`, `"url"`, `"effect"`, `"history"`, `"mixer"`, `"fill"`, `"stroke"`, `"swatches"`, `"layers"`, `"frames"`, `"behaviors"`, `"optimize"`, `"library"`, `"styles"`, `"optimized colors"`, `"options"`, and `"toolbox"`.

Returns

A rectangle that specifies the bounds of the panel (see “Rectangle” on page 6).

fw.getFloaterVisibility()

Availability

Fireworks 3

Description

Checks to see if a specified panel is visible.

Arguments

panelName

Acceptable values for *panelName* are `"find"`, `"project log"`, `"object"`, `"info"`, `"url"`, `"effect"`, `"history"`, `"mixer"`, `"fill"`, `"stroke"`, `"swatches"`, `"layers"`, `"frames"`, `"behaviors"`, `"optimize"`, `"library"`, `"styles"`, `"optimized colors"`, `"options"`, and `"toolbox"`.

Returns

`true` if the specified panel is visible; `false` otherwise.

fw.getHideAllFloaters()

Availability

Fireworks 3

Description

Returns the hidden or visible status of the panels.

Arguments

None.

Returns

`true` if the panels are hidden; `false` otherwise.

fw.getHTMLFileForScript()

Availability

Fireworks MX

Description

Returns an HTML file

Arguments

None.

Returns

A file URL.

fw.getNumberOfTables()

Availability

Fireworks MX

Description

Returns the number of top-level (that is, non-nested) tables in a document.

Arguments

`filename`

`filename` is the name of the file that contains the tables to be counted.

Returns

A long integer that represents the number of tables in the document.

fw.getPref()

Availability

Fireworks 3

Description

Returns the Preference value (string or numeric) that is associated with the specified Preference key.

Arguments

`prefkey`

`prefkey` is a string that specifies the Preference value to return. A complete list of these values is beyond the scope of this documentation, but the format of *`prefkey`* exactly matches that in the Fireworks Preferences file. To set a Preference value, use `fw.setPref()`.

Returns

A string or numeric Preference value.

fw.launchApp()

Availability

Fireworks MX

Description

Launches an application using a file URL that is returned by `fw.findApp()` on page 178. You can specify, optionally, files to open in the application.

Arguments

`appPath, filePathsToOpen`

`appPath` is a file URL that specifies the executable to launch. Typically, this value can be obtained by calling `fw.findApp()` on page 178.

`filePathsToOpen` is an array of file URLs to open in the executable to launch. It is safe to pass an empty array.

Returns

A Boolean value that indicates whether the application launched successfully.

fw.launchBrowserTo()

Availability

Fireworks MX

Description

Launches Fireworks' primary web browser to view a URL.

Arguments

`url`

`url` identifies the URL to open in the primary web browser.

Returns

Nothing.

Example

The following command launches a browser to view the Macromedia website:

```
fw.launchBrowserTo("http://www.macromedia.com");
```

fw.locateDocDialog()

Availability

Fireworks 4

Description

Displays a dialog box that lets the user choose one or more files. For syntax details, see "Using fw.locateDocDialog()" on page 20.

Arguments

`maxnumdocs, formatlist`

- *`maxnumdocs`* specifies the maximum number of documents to choose.
- *`formatlist`* is a list of acceptable file types to open.

Returns

An array of file:// URLs, or `null` if the dialog box is canceled.

fw.openDocument()

Availability

Fireworks 3, enhanced in 4

Description

Opens the specified file(s) in new document windows. If a file is already open, it opens again; to avoid redundant opens, call `findOpenDocument()` first.

Arguments

{fileURL}, *{bOpenAsNew}*

- *fileURL* is a string or an array of strings, where each is expressed as a file://URL, that specifies the file(s) to be opened. If *fileURL* is omitted or `null`, the Open Document dialog box appears.
- If *bOpenAsNew*, which was added in Fireworks 4, is `true`, the document(s) open as unsaved and untitled. If *bOpenAsNew* is `false` (the default value), they open with their original names.

Returns

If any of the file(s) can open, it returns the `Document` object for each file. Returns `null` if none of the documents can open.

fw.popupColorPicker()

Availability

Fireworks MX

Description

Opens the pop-up color swatches dialog to let the user visually select a color.

Arguments

screenLoc, initialColor, allowTransparent, forceWeb216

- *screenLoc* is the location at which the dialog appears, in the form of a point `{`*x:* `float`, *y:* `float}` (see "Point" on page 6 for syntax details).
- *initialColor* is the initially selected color in the dialog, in the form `#rrggbbaa` (see "Color string" on page 5 for syntax details).
- *allowTransparent* is a Boolean value that lets the user select a transparent color; set to `true` for transparent, `false` otherwise.
- *forceWeb216* is a Boolean value that forces the chosen color to fall within the web216 panel; set to `true` to force the color change, `false` otherwise.

Returns

The chosen color is formatted as `#rrggbbaa` (see "Color string" on page 5 for syntax details).

fw.popupColorPickerOverMouse()

Availability

Fireworks MX

Description

Opens the pop-up color swatches dialog at the current mouse location to let the user visually select a color.

Arguments

initialColor, allowTransparent, forceWeb216

- *initialColor* is a color string formatted as `#rrggbbaa` (see "Color string" on page 5 for syntax details), which is the initially selected color in the dialog.
- *allowTransparent* is a Boolean value that lets the user select a transparent color; set to `true` for transparent, `false` otherwise.

- *`forceWeb216`* is a Boolean value that forces the chosen color to fall within the web216 panel; set to `true` to force the color change, `false` otherwise.

Returns

The chosen color is formatted as `#rrggbbaa` (see "Color string" on page 5 for syntax details).

fw.quit()

Availability

Fireworks 4

Description

Identical to `fw.quitApplication()`.

fw.quitApplication()

Availability

Fireworks 3

Description

Quits Fireworks, but prompts the user to save any changed documents before exiting.

Arguments

None.

Returns

Nothing.

fw.readNthTable()

Availability

Fireworks MX

Description

Reads the table that the parameter indicates. The tables are zero-indexed.

Arguments

`filename, tablenumber`

`filename` is a *`fileURL`* for the file that contains the desired table.

`tablenumber` is a long integer that specifies the desired table; the tables are zero-indexed.

Returns

A database that is constructed from the table data.

fw.readPanelStateFromFile()

Availability

Fireworks MX

Description

Reads in a panel state file, which is generated by the `fw.writePanelStateToFile function`, and moves the panels, Property inspector, and toolbox to the appropriate locations.

Arguments

`filepath`

`filepath` is the location of the panel state file as a string in the format file://URL.

Returns

Nothing.

fw.replace()

Availability

Fireworks 3

Description

Verifies that the selection matches the current search string and replaces it with the replacement string.

Arguments

None.

Returns

The number of items that are replaced, or `-1` if there are items in the document that remain to be searched.

Related functions

`fw.setUpFindReplace()`

fw.replaceAll()

Availability

Fireworks 3

Description

Performs a Replace All operation on the active document using the current search-and-replacement strings.

Arguments

None.

Returns

The number of items replaced, or `-1` if the search is not yet complete.

Related functions

`fw.setUpFindReplace()`

fw.revertDocument()

Availability

Fireworks 3

Description

Reverts the specified document to its previously saved version.

Arguments

`{document}`

`document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to revert. If *`document`* is omitted or `null`, the active document reverts.

Returns

Nothing.

fw.runScript()

Availability

Fireworks 3

Description

Executes a JavaScript file.

Arguments

`filename`

`filename` is the name of the script file to execute. If *`filename`* is not a file URL (that is, it does not begin with `"file:///"`), it is assumed to be the name of a file in the Fireworks MX/ Configuration/Commands folder.

Returns

Result of script.

Example

The following command runs a script found in the Align Center to Document.jsf file, which is located in the Commands folder.

```
fw.runScript("Align Center to Document.jsf");
```

fw.saveAll()

Availability

Fireworks 3

Description

Saves all open documents, displaying the Save As dialog box for any documents that were not previously been saved.

Arguments

None.

Returns

Nothing.

fw.saveDocument()

Availability

Fireworks 3

Description

Saves the specified document as a native Fireworks PNG file with the specified name. To save a document to another format, such as GIF or JPEG, use `fw.exportDocumentAs()`.

Arguments

`document``, {`*`fileURL`*`}`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document to be saved. If *`document`* is `null`, the active document is saved.
- *`fileURL`* is the name of the saved document, which is expressed as a file://URL. If *`fileURL`* is `null` or omitted, the document is saved with its current name; if the document has not been saved, the Save As dialog box appears.

Returns

Nothing.

fw.saveDocumentAs()

Availability

Fireworks 3

Description

Displays the Save As dialog box for the specified document, so it can be saved as a native Fireworks PNG file with the specified name. To save a document to another format, such as GIF or JPEG, use `fw.exportDocumentAs()`.

Arguments

`document`

`document` is a `Document` object (for example, `fw.documents[2]`) that specifies the document to save. If *`document`* is `null`, the active document is saved.

Returns

The file URL for the saved document, or `null` if the dialog box was canceled.

fw.saveDocumentCopyAs()

Availability

Fireworks 3

Description

Saves a copy of the specified document as a native Fireworks PNG file with the specified name. To save a document to another format, such as GIF or JPEG, use `fw.exportDocumentAs()`.

Arguments

`document, fileURL`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies the document to be saved. If *`document`* is `null`, the active document is saved.
- *`fileURL`* is the filename for the saved file, which is expressed as a file://URL. If *`fileURL`* is `null`, the Save As dialog box appears.

Returns

The file URL for the saved document, or `null` if the dialog box was canceled.

fw.saveJsCommand()

Availability

Fireworks 3

Description

Saves the specified string of JavaScript code as a JSF command file.

Arguments

`jscode, filename`

- *`jscode`* specifies the string of code to be saved as a JSF command file.
- *`filename`* specifies the name in which the file should be saved. If *`filename`* is not a file URL (that is, it does not begin with `"file:///"`), the file is saved in the Fireworks MX/ Configuration/Commands folder.

Returns

Nothing.

fw.setActiveViewScale()

Availability

Fireworks MX

Description

Sets the zoom amount and the center of the view for the current document.

Arguments

`scale, center`

- *`scale`* is a floating-point number where 1.0 is 100 percent, or normal view.
- *`center`* is a point that defines the location in the document to center the view. This argument can be used to navigate around different parts of the document.

Returns

Nothing.

fw.setActiveWindow()

Availability

Fireworks 3

Description

Sets the specified document as the active document.

Arguments

`document, {trueFalse}`

- *`document`* is a `Document` object (for example, `fw.documents[2]`) that specifies which document should be made active.
- *`trueFalse`* (optional) is ignored by Fireworks. It is included only for Dreamweaver compatibility.

Returns

Nothing.

Example

The following command makes the fourth document the active document.

```
fw.setActiveWindow(fw.documents[3]);
```

fw.setFloaterGrouping()

Availability

Fireworks 3

Description

Moves the specified panel into another specified panel, changing it to a tab within that panel. This is the same behavior as dragging a tab from one panel to another or to its own panel.

Arguments

`panelNameToMove, panelNameToReceive`

- *`panelNameToMove`* is a lowercase string that specifies the panel to be moved.
- *`panelNameToReceive`* is a lowercase string that specifies the panel into which *`panelNameToMove`* should move. If *`panelNameToReceive`* is `null`, the *`panelNameToMove`* moves into its own panel.

Returns

Nothing.

Example

The following command moves the Stroke tab from its current location into the panel named Object. Although the panel name might be capitalized onscreen, it must be passed as lowercase.

```
fw.setFloaterGrouping("stroke", "object");
```

fw.setFloaterPosition()

Availability

Fireworks 3

Description

Sets the position and size of a panel.

Arguments

`panelName, boundingRectangle`

- Acceptable values for *`panelName`* are `"find"`, `"project log"`, `"object"`, `"info"`, `"url"`, `"effect"`, `"history"`, `"mixer"`, `"fill"`, `"stroke"`, `"swatches"`, `"layers"`, `"frames"`, `"behaviors"`, `"optimize"`, `"library"`, `"styles"`, `"optimized colors"`, `"options"`, and `"toolbox"`.
- *`boundingRectangle`* is a rectangle that specifies the size of the panel (see "Rectangle" on page 6). Some panels ignore the specified size but place the top left corner of the panel at the top left location of the specified rectangle.

Returns

Nothing.

fw.setFloaterVisibility()

Availability

Fireworks 3

Description

Shows or hides the specified panel.

Arguments

`panelName, bVisible`

- Acceptable values for *`panelName`* are `"find"`, `"project log"`, `"object"`, `"info"`, `"url"`, `"effect"`, `"history"`, `"mixer"`, `"fill"`, `"stroke"`, `"swatches"`, `"layers"`, `"frames"`, `"behaviors"`, `"optimize"`, `"library"`, `"styles"`, `"optimized colors"`, `"options"`, and `"toolbox"`.
- If *`bVisible`* is `true`, the specified panel is visible. If *`bVisible`* is `false`, the panel is hidden.

Returns

Nothing.

fw.setHideAllFloaters()

Availability

Fireworks 3

Description

Shows or hides the panels. This behavior is the same as the tab key functionality.

Arguments

`bHide`

If *`bHide`* is `true`, the panels are hidden. If *`bHide`* is `false`, the panels are visible.

Returns

Nothing.

fw.setPref()

Availability

Fireworks 3

Description

Sets the value that is associated with the specified Preference key.

Arguments

`prefname, prefval`

A complete list of these values is beyond the scope of this documentation, but the format of *`prefname`* and *`prefval`* exactly matches those in the Fireworks Preferences file. To return the value that is associated with a Preference key, use `fw.getPref()`.

Returns

Nothing.

fw.setUpFindReplace()

Availability

Fireworks 3

Description

Sets up a search.

Arguments

`findSpec`

`findSpec` is a `Find` object (see “Find” on page 15).

Returns

Nothing.

fw.toggleFloater()

Availability

Fireworks 3

Description

Shows, hides, or makes topmost the specified panel.

- If the panel is not visible, this function makes it visible and topmost.
- If the panel is topmost, this function hides it.
- If the panel is visible but not topmost, this function makes it topmost.

Arguments

`panelName`

Acceptable values for *`panelName`* are `"find"`, `"project log"`, `"object"`, `"info"`, `"url"`, `"effect"`, `"history"`, `"mixer"`, `"fill"`, `"stroke"`, `"swatches"`, `"layers"`, `"frames"`, `"behaviors"`, `"optimize"`, `"library"`, `"styles"`, `"optimized colors"`, `"options"`, and `"toolbox"`.

Returns

Nothing.

fw.ungroupPrimitives()

Availability

Fireworks 4

Description

Replaces selected primitive objects with their equivalent paths. The new objects have all the attributes of the ones they replaced (mask, stroke, fill, and so on).

Arguments

None.

Returns

Nothing.

Related functions

`dom.addNewRectanglePrimitive()`

fw.updateHTML()

Availability

Fireworks 4

Description

Updates the HTML that was previously exported from Fireworks.

Arguments

`doc`, *`htmlUrl`*, *`bRecoverFromError`*

- *`doc`* is a `Document` object that specifies the document to be used for updating the HTML (see "Document" on page 9). If *`doc`* is `null`, the active document is used.
- *`htmlUrl`* is the filename for the HTML file to update, which is expressed as a file://URL. To force Fireworks to display the Update HTML dialog box, pass `null` for *`htmlUrl`*. If you pass `null` for *`htmlUrl`*, *`bRecoverFromError`* is ignored.
- If *`bRecoverFromError`* is `true` and the HTML update encounters an error, Fireworks displays a Confirmation dialog box and attempts to recover. If it is `false`, Fireworks fails without notifying the user if it encounters an error.

Returns

`true` if the HTML was updated; `false` otherwise.

Example

The following command updates the images in an HTML file, using the current document.

```
fw.updateHTML(null, "file:///C|/mysite/nav.htm", true);
```

fw.writePanelStateToFile()

Availability

Fireworks MX

Description

Writes out the panel states (location, size, open or closed, and so on), toolbox state, and Property inspector state to an XML file that is specified by the argument.

Arguments

`filepath`

`filepath` is a string that identifies the destination XML file in the format file://URL.

Returns

Nothing.

fw.yesNoDialog()

Availability

Fireworks MX

Description

Prompts the user with a dialog box that contains buttons that are labeled Yes and No.

Arguments

`promptString`

`promptString` is the prompt message that appears in the dialog box.

Returns

A Boolean value: `true` if the user selected the Yes button; `false` otherwise.

Example

```
var shouldDuplicate = fw.yesNoDialog("Would you like to duplicate the
  element?");
```

Property inspector functions

These functions control the Properties window, which shows details about the current document or selected object.

fw.showPIWindow()

Availability

Fireworks MX

Description

Opens the Property inspector window.

Arguments

None.

Returns

Nothing.

fw.hidePIWindow()

Availability

Fireworks MX

Description

Makes the Property inspector window invisible.

Arguments

None.

Returns

Nothing.

fw.isPIExpanded()

Availability

Fireworks MX

Description

Returns the current expanded state of the Property inspector (expanded or minimized).

Arguments

None.

Returns

A Boolean value: `true` if expanded; `false` otherwise.

fw.isPIVisible()

Availability

Fireworks MX

Description

Returns the current visible state of the Property inspector (hidden or shown).

Arguments

None.

Returns

A Boolean value: `true` if visible; `false` otherwise.

fw.growPIWindow()

Availability

Fireworks MX

Description

Sets the Property inspector window to its expanded state.

Arguments

None.

Returns

Nothing.

fw.shrinkPIWindow()

Availability

Fireworks MX

Description

Sets the Property inspector window to its minimized state.

Arguments

None.

Returns

Nothing.

fw.setPIPosition()

Availability

Fireworks MX

Description

Moves the top-left corner of the Property inspector window to the specified location.

Arguments

`pt`

`pt` is a point that is given in screen coordinates.

Returns

Nothing.

fw.getPIPosition()

Availability

Fireworks MX

Description

Retrieves the location of the top-left corner of the Property inspector in screen coordinates.

Arguments

None.

Returns

A point object that is formatted as `{`*x:* `float`, *y:* `float}` (see "Point" on page 6 for syntax details), which contains the location of the Property inspector.

History panel functions

These functions control the History panel.

fw.historyPalette.clearSteps()

Availability

Fireworks 3

Description

Clears the undo and redo stack.

Arguments

None.

Returns

Nothing.

fw.historyPalette.copySteps()

Availability

Fireworks 3

Description

Copies the selected history steps to the Clipboard.

Arguments

`array of indexes`

`array of indexes` is a zero-based array that specifies which steps from the History panel should be copied. If it is `null`, the currently selected steps are used.

Returns

Nothing.

fw.historyPalette.getSelection()

Availability

Fireworks 3

Description

Determines which steps in the History panel are selected.

Arguments

None.

Returns

A zero-based array that represents which History panel steps are selected.

fw.historyPalette.getStepCount()

Availability

Fireworks 3

Description

Returns the number of steps in the History panel.

Arguments

None.

Returns

The number of steps in the History panel (not a zero-based value).

fw.historyPalette.getStepsAsJavaScript()

Availability

Fireworks 3

Description

Gets the JavaScript equivalent of the selected steps.

Arguments

`array of indexes`

`array of indexes` is a zero-based array that specifies which steps from the History panel should be returned as JavaScript. If the argument is `null`, the currently selected steps are returned.

Returns

A JavaScript string.

Related functions

`fw.historyPalette.replaySteps()`

fw.historyPalette.getUndoState()

Availability

Fireworks 3

Description

Returns a string that indicates the current undo state to be used for later calls to `fw.historyPalette.setUndoState()`.

Arguments

None.

Returns

The string to use with `fw.historyPalette.setUndoState()`. This string is designed to be used internally by Fireworks only and might change format in the future. Do not try to parse this string or construct a custom string to pass to `fw.historyPalette.setUndoState()`.

fw.historyPalette.replaySteps()

Availability

Fireworks 3

Description

Gets the JavaScript equivalent of the selected steps and executes them.

Arguments

`array of indexes`

`array of indexes` is a zero-based array that specifies which steps from the History panel should be returned as JavaScript and executed. If the argument is `null`, the currently selected steps are used.

Returns

A JavaScript string.

Related functions

`fw.historyPalette.getStepsAsJavaScript()`

fw.historyPalette.saveAsCommand()

Availability

Fireworks 3

Description

Gets the JavaScript equivalent of the selected steps and saves them as a JSF command file.

Arguments

`array of indexes, {filename}`

- *`array of indexes`* indicates which steps from the History panel should be saved. For example, to save the first, third, and sixth steps in the History panel, pass `[0, 2, 5]`. If this argument is `null`, the currently selected steps are used.
- *`filename`* is an optional string that specifies a name for the JSF command file. It can be any string, including a file:// URL. If *`filename`* is omitted or `null`, the user is prompted for a filename. If *`filename`* is not a file://URL, the file is saved in the Fireworks MX/Configuration/Commands folder with the specified filename.

Returns

Nothing.

fw.historyPalette.setSelection()

Availability

Fireworks 3

Description

Sets the portion of the History panel that is selected.

Arguments

`array of indexes`

`array of indexes` specifies which steps in the History panel are selected. Values are zero-based. For example, to select the first, third, and sixth steps in the History panel, pass `[0, 2, 5]`.

Returns

Nothing.

fw.historyPalette.setUndoState()

Availability

Fireworks 3

Description

Performs the correct number of undo or redo operations to arrive at the selected state.

Arguments

`undoStateString`

`undoStateString` is the string that `fw.historyPalette.getUndoState()` returns.

Returns

Nothing.

Using the common API

To enable commands to use a common syntax (and perhaps the ability to run a single command in multiple applications), a common Macromedia API exists. You can access this API using `app.methodName()`.The following methods are currently supported in Fireworks and Dreamweaver to let developers easily create commands for both applications.

app.toggleFloater()

Identical to "fw.toggleFloater()" on page 193.

app.setFloaterVisibility()

Identical to "fw.setFloaterVisibility()" on page 191.

app.getRootDirectory()

Identical to "`appDir` •" on page 17.

app.browseDocument()

Identical to "fw.browseDocument()" on page 168.

Note: app.getRootDirectory() is helpful for using app.browseDocument() to view files within the applications's folder.

Using the addBehavior() function

The following code shows the syntax for `dom.addBehavior()`:

`fw.getDocumentDOM().addBehavior(`*`action, event, eventindex`*`);`

The first argument is a string that specifies the behavior to be added (see "dom.addBehavior()" on page 58). The information in this section describes the acceptable values for the first argument that is passed to `dom.addBehavior()`.

MM_nbGroup [down]

Availability

Fireworks 3

Description

Sets a navigation bar "down" behavior.

Arguments

`type, barName, target, swapFrame, fileName, preload`

- Pass `"down"` for *`type`*.
- Pass `"navbar1"` for the name of the navigation bar.
- *`target`* specifies the slice to which the behavior is attached. Pass `-1` for this value; all other values are used internally by Fireworks.
- *`swapFrame`* is a zero-based integer that specifies the frame to swap. To use *`fileName`* as a URL, pass `-1` here.

- *fileName* specifies the frame or file to swap. If you specified a frame to use in *swapFrame*, pass an empty text string. If you want to specify a filename and you passed `-1` for *swapFrame*, pass the string for the relative URL of the image.
- *preload* is a binary value that specifies whether to preload the swapped image (pass `1`) or not (pass `0`).

Example

```
fw.getDocumentDOM().addBehavior("MM_nbGroup(\’down\’,\’navbar1\’,-
  1,2,\"\",1)", "onClick", -1);
```

MM_nbGroup [highlight]

Availability

Fireworks 3

Description

Sets a navigation bar highlight behavior.

Arguments

type, *target*, *swapFrame*, *fileName*, *preload*, *downHighlight*, *downHighlightFrame*, *downHighlightFilename*

- Pass `"over"` for *type*.
- *target* specifies the slice to which the behavior is attached. Pass `-1` for this value; all other values are used internally by Fireworks.
- *swapFrame* is a zero-based integer that specifies the frame to swap. To use *fileName* as a URL, pass –1 here.
- *fileName* specifies the frame or file to be swapped. If you specified a frame to use in ***swapFrame***, pass an empty text string. If you want to specify a filename and you passed `-1` for *swapFrame*, pass the string for the relative URL of the image.
- *preload* is a binary value that specifies whether to preload the swapped image (pass 1) or not (pass `0`).
- *downHighlight* is a binary value that specifies whether an image should be used for highlighting on mouse down (pass `1`) or not (pass `0`). If you pass `1`, use the next two arguments to specify the frame or image to be used.
- *downHighlightFrame* is a zero-based integer that specifies the frame to use as a highlight image. To use *downHighlightFrame* as a URL, pass `-1` here.
- *downHighlightFilename* specifies the frame or file to be used as the highlight image. If you specified a frame to use in *downHighlightFrame*, pass an empty text string. If you want to specify a filename and you passed `-1` for *downHighlightFrame*, pass the string for the relative URL of the image.

Example

```
fw.getDocumentDOM().addBehavior("MM_nbGroup(\'over\',-1,1,\"\",1,0,3,\"\")",
  "onMouseOver", -1);
```

MM_nbGroup [image]

Availability

Fireworks 3

Description

Sets a navigation bar image behavior.

Arguments

`type, downHighlight, initiallyDown`

- Pass `"all"` for *`type`*.
- *`downHighlight`* is a binary value that specifies whether the image should be highlighted on a mouse down action (pass `1`) or not (pass `0`).
- *`initiallyDown`* is a binary value that specifies whether the image should initially appear as in the “down” state (pass `1`) or not (pass `0`).

Example

```
fw.getDocumentDOM().addBehavior("MM_nbGroup(\'all\',1,0)", "onMouseOver", -1);
```

MM_nbGroup [out]

Availability

Fireworks 3

Description

Sets a navigation bar restore behavior.

Arguments

`type`

Pass `"out"` for *`type`*.

Example

```
fw.getDocumentDOM().addBehavior("MM_nbGroup(\'out\')", "onMouseOut", -1);
```

MM_simpleRollover

Availability

Fireworks 3

Description

Adds a simple rollover behavior.

Arguments

None.

Example

```
fw.getDocumentDOM().addBehavior("MM_simpleRollover()", "onMouseOver", -1);
```

MM_statusMessage

Availability

Fireworks 3

Description

Sets a status bar message.

Arguments

`message`

`message` is a string that specifies the status message to appear.

Example

```
fw.getDocumentDOM().addBehavior("MM_statusMessage(\"Status Message!\")",
  "onMouseOver", -1);
```

MM_swapImage

Availability

Fireworks 3

Description

Adds a swap image behavior.

Arguments

`target, swapFrame, fileName, preload, restoreOnMouseOut`

- *`target`* specifies the slice to which the behavior is attached. Pass `-1` for this value; all other values are used internally by Fireworks.
- *`swapFrame`* is a zero-based integer that specifies the frame to swap. To use *`fileName`* as a URL, pass `-1` here.
- *`fileName`* specifies the frame or file to swap. If you specified a frame to use in *`swapFrame`*, pass an empty text string. If you want to specify a filename and you passed `-1` for *`swapFrame`*, pass the string for the relative URL of the image.
- *`preload`* is a binary value that specifies whether to preload the swapped image (pass `1`) or not (pass `0`).
- *`restore`* is a binary value that specifies whether to restore on a mouse out action (pass `1`) or not (pass `0`).

Example

```
fw.getDocumentDOM().addBehavior("MM_swapImage(-1,1,\"\",1,1)", "onMouseOver",
  -1);
```

MM_swapImgRestore

Availability

Fireworks 3

Description

Adds a swap image restore behavior.

Arguments

None.

Example

```
fw.getDocumentDOM().addBehavior("MM_swapImgRestore()", "onMouseOut", -1);
```

Using Macromedia Flash to create custom panels and commands

Fireworks MX contains a special Macromedia Flash reader that lets Shockwave files (SWFs) play as panels and commands in the Fireworks interface. Additionally, developers can install a Macromedia API wrapper extension for Macromedia Flash to facilitate creating SWFs that communicate with the Fireworks API. By leveraging the new API communication between Macromedia Flash and Fireworks, Fireworks extension developers can create interfaces and dialog boxes for their commands that go beyond the `alert()` and `prompt()` dialog boxes that are supported in previous versions. You can add command panels to Fireworks MX for image enhancements, object manipulation, or for other custom functionality.

How custom panels and commands work

Macromedia Flash developers can create interactive movies that contain a combination of ActionScript and calls to the Fireworks API for two types of deployment: interactive panels or modal commands. Basically, while writing ActionScript, a Macromedia Flash developer can embed commands for the Fireworks API in the `MMExecute()` function, or by using the API wrapper extension for Macromedia Flash. These Macromedia Flash animations can be constructed as interactive panels that work the same as built-in panels, such as the Layers panel or the Frames panel.

SWFs that are published to the Fireworks MX installation directory, Configuration\Command Panels subfolder act as panels in the Fireworks interface at runtime and are available through the Window menu.

SWFs that are published to the Configuration\Commands subfolder act as modal commands and are available through the Commands menu in the Fireworks interface.

Note: On multiuser systems, Fireworks supports a Command Panels folder inside of each user's Configuration folder, so users can save favorite panels.

At runtime, Fireworks has a special Macromedia Flash player that runs the SWF animations, or commands, as the user clicks on the custom command options, which is similar to the Window > Align panel.

Developing Fireworks panels and commands in Macromedia Flash

Any part of the Fireworks API can be called by embedding them in the following Macromedia Flash API functions. These functions communicate directly with the special Macromedia Flash player that is distributed with Fireworks MX:

MMExecute()

Description

Declares a set of JavaScript to pass to the Fireworks API, allowing Flash authors to embed Fireworks API commands in a frame of a Flash movie.

Note: `MMExecute` supersedes the `FWJavascript` command. However, the `FWJavascript` command still works in the current version of Fireworks.

Arguments

`jsToPass`

`jsToPass` is a string of JavaScript for Fireworks to execute.

Returns

Nothing.

Example

```
MMExecute("fw.getDocumentDOM().addNewRectanglePrimitive({left:47, top:26,
  right:102, bottom:87}, 0");
```

MMEndCommand()

Description

This function should be called by whatever OK or Cancel buttons that the Macromedia Flash author provides to the user to execute a command (only for modal commands, not for Flash panels).

Note: `MMEndCommand` supersedes the `FWEndCommand` command. However, `FWEndCommand` still works in the current version of Fireworks.

Arguments

`endStatus`, *`notifyString`*

- *`endStatus`* is a Boolean value: `true` to commit changes; `false` otherwise. If it is `false`, any changes the command or panel might have made to the document are discarded. To commit the changes, *`endStatus`* must be `true`.
- *`notifyString`* is a string to notify the user of errors; use only if you pass "`false`" for the first argument. For OK, pass an empty string.

Returns

Nothing.

Using the API wrapper extension in Macromedia Flash

You can install a special extension that was developed specifically for writing Fireworks functions into ActionScript either as a replacement for, or in conjunction with, using `MMExecute()` and `MMEndCommand()`. After it is installed, the API wrapper appears in the Macromedia Flash interface to make writing commands for Fireworks easier. Instead of having to embed every Fireworks function in `MMExecute()`, you can use a series of `fwapi` functions in the ActionScript. Then, when it is published, the wrapper translates the `fwapi` functions into the expanded Fireworks functions. You can also mix the `fwapi` functions with `MMExecute()` statements.

To install the API wrapper, make sure you have the Macromedia Extensions Manger installed and double-click on the Extension file. In Macromedia Flash, the wrapper appears in the Components window as FWCommandComponents.

The following example shows a command without the wrapper:

```
var path = MMExecute("fw.appPatternsDir;");
```

The following example shows the same command using the wrapper:

```
var path =fwapi.getAppPatternsDir();
```

Working with AS files

Keeping a separate .as file for the ActionScript allows for easier edits later without having to open and edit the FLA file directly. Your FLA file needs to have a `#include "myStringFile.as"` in the first frame (where "myStringFile" is the actual name of your AS file) so the ActionScript strings are complied at publishing time.

Note: The FLA files and the AS files should reside in the same folder so that there is no problem finding the AS file for compiling.

Guidelines for creating panels and commands

Nested quotation marks need to use the backslash convention (\). The following example prints: `John's example is really "complex"!`

```
MMExecute('alert("John\'s example is really \"complex\"!")';
```

- The movie size set in Flash is used in Fireworks as the minimum and default size for the command panel.
- To improve the appearance and positioning of a modeless panel, turn off scaling and align the panel contents to the top-left corner of the stage. You can make these changes with the following ActionScript:

```
Stage.align = "TC";
Stage.scaleMode = "noScale";
```

Publishing

When testing your script, use the File > Publish menu option in Macromedia Flash MX. The SWF file is in the same place as the FLA file after publishing.

Debugging

Use the following functions to show or hide everything that the SWF passes to the Fireworks API during execution. Place these debug functions around the suspect code in your Macromedia Flash ActionScript to turn the debug functions on or off as needed. Be careful to use these functions only around "suspect" code; otherwise, you might encounter a long series of dialog statements.

Note: The debugging commands work even if you are running a .jsf file.

fw.enableFlashDebugging()

Availability

Fireworks MX

Description

Turns on debug messages for Flash commands. When Flash debugging is enabled, every time a Flash command calls `MMExecute()`, Fireworks displays the command string in a dialog box. This function is particularly useful for monitoring which commands are executed in a command panel.

Arguments

None.

Returns

Nothing.

fw.disableFlashDebugging()

Availability

Fireworks MX

Description

Turns off debug messages for Flash commands. See "fw.enableFlashDebugging()" on page 208 for a description of the Flash debugging capabilities.

Arguments

None.

Returns

Nothing.

INDEX