Extending
ADOBE® FIREWORKS®
Legal notices

# Contents

## Chapter 1: Extending Fireworks Overview
- Prerequisites ................................................................. 1
- Installing an extension ..................................................... 1
- What's new in Extending Fireworks CS5 ............................. 2
- Conventions used in this guide ......................................... 6
- Additional resources for extension writers ......................... 6

## Chapter 2: The Fireworks Object Model
- Using the Fireworks Object Model .................................... 7
- Using Fireworks API functions ......................................... 9
- Using the common API .................................................... 10
- Working with selected objects .......................................... 10
- Global methods ............................................................ 14
- Core objects ..................................................................... 17
- Adobe AIR Extension ...................................................... 22

## Chapter 3: The Document object
- Document object properties ............................................. 28
- Document functions ......................................................... 31

## Chapter 4: The Fireworks Object
- Fireworks functions .......................................................... 194

## Chapter 5: Objects within Fireworks documents
- Behavior object .............................................................. 243
- Brush object ..................................................................... 243
- Contour object ............................................................... 246
- ContourNode object ......................................................... 246
- ContourNodeDynamicInfo object .................................... 247
- ControlPoint object ........................................................ 247
- Effect object ..................................................................... 248
- EffectList object ............................................................ 254
- Element object ............................................................... 254
- ElementMask object ........................................................ 259
- ExportFrameInfo object .................................................. 260
- ExportOptions object ...................................................... 260
- ExportPaletteInfo object ................................................ 262
- ExportSettings object ...................................................... 263
- Fill object ........................................................................ 265
- Frame object .................................................................... 266
- FrameNLayerIntersection object ..................................... 266
- Gradient object .............................................................. 267
- GradientNode object ........................................................ 267
- Guides object .................................................................... 267

Last updated 5/17/2012
Chapter 1: Extending Fireworks Overview

To extend Adobe® Fireworks® CS5, you must write JavaScript code. You can use JavaScript to write your own objects 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 guide introduces the Fireworks Object Model, explains how to write cross-product extensions (extensions written in, or for, other Adobe applications), and discusses the JavaScript Auto Shape construction. The final chapter is a reference to 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 guide 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 guide 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).

Installing an extension

As you start learning the process of writing extensions, you might want to explore the extensions and resources already available through the Adobe Exchange website (www.adobe.com/go/exchange). By installing an existing extension, you will become familiar with some of the tools that you need to work with your own extensions.

To install an extension:

1. Download and install the Adobe® Extension Manager, which is available on the Adobe Downloads website (www.adobe.com/exchange/em_download/).
2. Log on to the Adobe Exchange website (www.adobe.com/go/exchange).
3. Click the Fireworks Exchange link.
4. From the available extensions, choose one that you want to use. Click the Download link to download the extension package.
5. Save the extension package in a directory on your machine.
6. In Fireworks, choose Commands > Manage Extensions to start the Extension Manager (or you can start Extension Manager, located in the Adobe program group, independently from Fireworks).
7. In the Extension Manager, choose File > Install Extension, and choose the extension package you just saved. The Extension Manager automatically installs the extension into Fireworks.
You cannot begin using some extensions until you restart Fireworks. If you are running Fireworks when you install the extension, you might be prompted to quit and restart the application.

To view basic information on the extension after its installation, go to the Extension Manager (Commands > Manage Extensions) in Fireworks.

**What’s new in Extending Fireworks CS5**

Fireworks CS5 includes the following new features and interfaces that you can use to develop extensions for the product:

**CSS-based layouts**  Fireworks CS5 now allows you to design complete web pages and export web standards-compliant CSS-based layouts. You can select a layout and integrate foreground and background graphics with automatic margin and padding detection. You can drop HTML rich symbols on your Fireworks layouts to specify headings, links, and form properties for precise CSS control. You can also use rectangles with rounded corners and gradients for creating layouts.

**Adobe Type Engine**  The new Adobe Type Engine provides enhanced typesetting capabilities similar to Adobe® Photoshop® and Adobe® Illustrator®. You can import or copy/paste double-byte characters from Illustrator or Photoshop without loss of fidelity, and create advanced effects such as setting text inside a path.

**Export to PDF**  You can generate high-fidelity, interactive, secure PDF documents from your Fireworks design comps and receive review comments through a shared PDF review.

**Smart Guides**  Use Smart Guides to expertly position and layouts objects on canvas. Use Smart Guides with tool tips to achieve precise positioning.

**Live Styles**  With Live Styles, you can use professionally designed styles or create your own collection. Enhanced capabilities include redefining styles, dynamically updating effects, colors, and text attributes by updating the style source.

**FXG export**  You can export designs as FXG files and use them in advanced design tools to develop rich Internet applications and experiences that can run on Adobe® Flash Player or Adobe® AIR®.

**Adobe AIR export**  You can package your Fireworks documents as Adobe AIR applications that you can secure using a digital signature.

The following APIs have been added or modified for this release:
## Extending Fireworks Overview

### Class

<table>
<thead>
<tr>
<th>Property or Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AIRExt</strong></td>
<td></td>
</tr>
<tr>
<td>AIRext.PreviewDocument()</td>
<td>Calls the ADL tool to preview an HTML document as an Adobe AIR application.</td>
</tr>
<tr>
<td>AIRext.PackageApplication()</td>
<td>Calls the ADT tool package an Adobe AIR application.</td>
</tr>
<tr>
<td>AIRext.CreatePackage()</td>
<td>Calls the ADT tool to create a signed AIR package.</td>
</tr>
<tr>
<td>AIRext.CheckCertificatePassword()</td>
<td>Calls the ADT tool to check a certificate password.</td>
</tr>
<tr>
<td>AIRext.SaveDigSigPassword()</td>
<td>Saves the password for the given digital certificate, for the current session of Fireworks (no persistent storage).</td>
</tr>
<tr>
<td>AIRext.GetDigSigPassword()</td>
<td>Returns the name of the generated password file.</td>
</tr>
<tr>
<td>AIRext.GetLastErrorLogName()</td>
<td>Returns the name of the generated error log file if one was created.</td>
</tr>
<tr>
<td>AIRext.GetAIRInstallPath()</td>
<td>Returns the full system path name and path of the generated file.</td>
</tr>
<tr>
<td>AIRext.GetJREVersion()</td>
<td>An object with two properties - maxVersion and minVersion; if an error occurs the values of the maxVersion and minVersion will be -1.</td>
</tr>
<tr>
<td>AIRext.CreateCertificate()</td>
<td>Calls the ADT to create a self-signed digital certificate.</td>
</tr>
<tr>
<td><strong>exportDoc</strong></td>
<td></td>
</tr>
<tr>
<td>isBackground</td>
<td>Specifies whether the document has a background image. If an image is set in the HTML set up, returns true.</td>
</tr>
<tr>
<td>backgroundAttachment</td>
<td>Specifies whether the document is fixed or scrolls. Values are Fixed or scroll.</td>
</tr>
<tr>
<td>docAlignment</td>
<td>Specifies the page alignment as left, center, right. Default is left.</td>
</tr>
<tr>
<td>backgroundURL</td>
<td>Specifies the URL of the background image as file:///</td>
</tr>
<tr>
<td>backgroundRepeat</td>
<td>Specifies if the background image repeats, and if repeats in X, Y, Repeat (Both) and No Repeat</td>
</tr>
<tr>
<td>leftMargin</td>
<td>Specifies the left margin of the page.</td>
</tr>
<tr>
<td>topMargin</td>
<td>Specifies the top margin of the page.</td>
</tr>
<tr>
<td>rightMargin</td>
<td>Specifies the right margin of the page.</td>
</tr>
<tr>
<td>bottomMargin</td>
<td>Specifies the bottom margin of the page.</td>
</tr>
</tbody>
</table>
### Extending Fireworks Overview

<table>
<thead>
<tr>
<th>Class</th>
<th>Property or Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SliceInfo</td>
<td>backgroundHorizontalPosition</td>
<td>Horizontal position of the background image. (Left, right, or center, numeric value)</td>
</tr>
<tr>
<td></td>
<td>backgroundVerticalPosition</td>
<td>Horizontal position of the background image. (Left, right, or center, numeric value)</td>
</tr>
<tr>
<td></td>
<td>backgroundHorizontalPositionVal</td>
<td>Horizontal position value of the background image if the argument is value.</td>
</tr>
<tr>
<td></td>
<td>backgroundVerticalPositionVal</td>
<td>Vertical position value of the background image if the argument is value.</td>
</tr>
<tr>
<td></td>
<td>isBackground</td>
<td>Specifies whether the slice is background image or foreground image. True = Background.</td>
</tr>
<tr>
<td></td>
<td>backgroundRepeat</td>
<td>Specifies if the background image repeats, and if repeats in X, Y, Repeat (Both) and No Repeat.</td>
</tr>
<tr>
<td></td>
<td>backgroundAttachment</td>
<td>Specifies if the bg image is fixed or scrolls with the rest of the page.</td>
</tr>
<tr>
<td>RectPrimitive</td>
<td>Mode</td>
<td>Specifies the roundness units which can be relative (percentage) or exact (pixels).</td>
</tr>
<tr>
<td>instance</td>
<td>transform</td>
<td>Specifies the transformation matrix of a symbol instance.</td>
</tr>
<tr>
<td></td>
<td>symbolName</td>
<td>Specifies the symbol name.</td>
</tr>
<tr>
<td>frame</td>
<td>name</td>
<td>Species the name of the state.</td>
</tr>
<tr>
<td>FrameNLayerIntersection</td>
<td>name</td>
<td>Specifies the name of the layer when the layers are accessed from states.</td>
</tr>
<tr>
<td>document</td>
<td>dom.setRectRoundnessMode()</td>
<td>Specifies the mode of corner roundness of the rectangle as either a percentage value or or exact pixel value.</td>
</tr>
<tr>
<td></td>
<td>dom.exportElements()</td>
<td>Exports an array of elements on the canvas to a 32-bit PNG image, based on the image export settings.</td>
</tr>
<tr>
<td></td>
<td>dom.moveNineScaleGuide()</td>
<td>Moves a 9-slice scaling guide’s position by specified pixels.</td>
</tr>
<tr>
<td></td>
<td>dom.placeTextInPath()</td>
<td>Places the selected text inside the selected path. If no text and path are selected, no action occurs.</td>
</tr>
<tr>
<td></td>
<td>dom.setSliceType()</td>
<td>Sets the slice type as image (foreground image), background image, or empty (HTML Slice).</td>
</tr>
<tr>
<td></td>
<td>dom.redefineStyleByName()</td>
<td>Redefines the target style with the source style.</td>
</tr>
<tr>
<td></td>
<td>dom.snapToPixel()</td>
<td>Snap To Pixel is applied on the selected object on the canvas.</td>
</tr>
<tr>
<td></td>
<td>dom.attachTextInPath()</td>
<td>Attaches the selected text inside the selected path. If no text and path are selected, no action occurs.</td>
</tr>
</tbody>
</table>
## Extending Fireworks Overview

<table>
<thead>
<tr>
<th>Class</th>
<th>Property or Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fw</td>
<td>fw.saveAs()</td>
<td>Saves the specified document in the specified filename and format.</td>
</tr>
<tr>
<td></td>
<td>fw.exportPDF()</td>
<td>Exports the specified document to the specified file in PDF format.</td>
</tr>
<tr>
<td></td>
<td>fw.exportFXG()</td>
<td>Exports current page, all pages of the open document in Fireworks, or selected objects from the current page as FXG files.</td>
</tr>
<tr>
<td></td>
<td>fw.exportPages()</td>
<td>Exports the specified pages in the specified format.</td>
</tr>
<tr>
<td></td>
<td>fw.shrinkPIWindow()</td>
<td>Minimizes the Property inspector window.</td>
</tr>
<tr>
<td></td>
<td>fw.setUseAGMRenderingForSelection()</td>
<td>Sets the currently selected vectors to use the new AGM Rendering.</td>
</tr>
<tr>
<td></td>
<td>fw.currentWorkspaceName()</td>
<td>Returns the name of the current workspace layout.</td>
</tr>
<tr>
<td></td>
<td>fw.getFamilyNameForPSFont()</td>
<td>Gets the family name of a PostScript font.</td>
</tr>
<tr>
<td></td>
<td>fw.getPlatformNameForPSFont()</td>
<td>Returns the platform name of the PostScript font.</td>
</tr>
<tr>
<td></td>
<td>“fw.saveDocumentAsTemplate(arg1,arg2)” on page 233</td>
<td>Saves the current document as a template.</td>
</tr>
<tr>
<td></td>
<td>“fw.saveDocumentAsTemplateAsync(arg1,arg2)” on page 234</td>
<td>Works the same as fw.saveDocumentAsTemplate() except for the async mode of save operation.</td>
</tr>
<tr>
<td></td>
<td>“fw.newDocumentFromTemplate(arg1)” on page 224</td>
<td>Opens up a file for selecting any template from the pre-defined Templates folder in Fireworks installation directory.</td>
</tr>
<tr>
<td></td>
<td>“fw.newMobileDocument(arg1,arg2,arg3,arg4)” on page 224</td>
<td>Creates a mobile document with the width, height, and resolution details.</td>
</tr>
<tr>
<td></td>
<td>“fw.previewInDeviceCentral()” on page 227</td>
<td>Launches Adobe Device Central and previews the current document in it. If there is no active document, an alert message is generated.</td>
</tr>
<tr>
<td></td>
<td>“fw.getDocumentDOM().insertPageForImport” on page 214</td>
<td>Creates a compound shape from the selected vectors applying the operation that is specified. Requires two or more vectors to be selected.</td>
</tr>
<tr>
<td></td>
<td>“fw.getDocumentDOM().combineCompoundShape” on page 213</td>
<td>Combines the compound shape that is selected. Requires selection of a compound shape.</td>
</tr>
<tr>
<td></td>
<td>“fw.getDocumentDOM().createCompoundShape” on page 214</td>
<td>Creates a compound shape from the selected vectors applying the operation that is specified. Requires two or more vectors to be selected.</td>
</tr>
<tr>
<td></td>
<td>fw.appTemplatesDir (See “The Fireworks Object” on page 191)</td>
<td>Returns the path of the Templates directory present at the location where you have installed Fireworks.</td>
</tr>
<tr>
<td></td>
<td>“fw.getDocumentDOM().VectorOperation” on page 215</td>
<td>Vector operation is applied on paths created after applying this API. A compound shape is created.</td>
</tr>
</tbody>
</table>

### Files

|          | writeUTF8          | Writes text in UTF8 encoding when the file is opened in UTF8 encoding. |
|          | readLineUTF8       | Reads one line from the file that has been opened with UTF8 encoding. |
Conventions used in this guide

The following typographical conventions are used in this guide:

- **Code** font indicates code fragments and API literals, including class names, method names, function names, type names, scripts, SQL statements, and HTML and XML tag and attribute names.

- **Italic code** font indicates replaceable items in code.

- The continuation symbol (\_) indicates that a long line of code has been broken across two or more lines to fit on the page. When copying the lines of code, eliminate the continuation symbol and type the code as one line.

- Curly braces ({ }) around a function argument indicate that the argument is optional.

  The following naming conventions are used in this guide:

- You refers to the developer who is responsible for writing extensions.

- The user refers to the person using Fireworks.

- The visitor refers to the person who views the graphic that the user created.

Additional resources for extension writers

To communicate with other developers who are writing extensions, you can visit the Adobe online forums at [www.adobe.com/support/forums/](http://www.adobe.com/support/forums/).
Chapter 2: The Fireworks Object Model

If you want to extend the functionality of Adobe Fireworks CS5 by writing or modifying a JavaScript extensibility file, you must become familiar with the objects that Fireworks makes available through JavaScript. The hierarchy of these objects comprises the Fireworks Object Model, which contains the following major components:

- Six global methods that are available from any part of the application and need not be declared as methods of a particular object. For more information, see “Global methods” on page 14.
- Core objects: Dialogs, Document, pngText, Errors, Files, Find, and System. For more information, see “Core objects” on page 17 and “The Document object” on page 28. (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 (for more information, see “The Fireworks Object” on page 191).
- Numerous objects associated with Fireworks documents, such as ExportOptions, Guides, Path, Image, and Text. For more information, see “Objects within Fireworks documents” on page 243.
- A set of objects that you can use to specify the format of HTML code when exporting from Fireworks. For more information, see “HTML export objects” on page 277.

Using the Fireworks Object Model

When scripting extensions for Fireworks, you write JavaScript commands that send calls to the Fireworks Object Model to determine or change the current settings for a Fireworks document. For example, the following command calls the Fireworks object (fw) to obtain the path to the Export Settings directory (appExportSettingsDir), which is expressed as a file://URL. In other words, fw references the Fireworks global object, of which appExportSettingsDir is a property (for more information, see “The Fireworks Object” on page 191), so a JavaScript command can assign the resulting value to a variable, as follows:

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

Accessing a Fireworks document

All the functions listed in “Property inspector functions” on page 321 are methods of the Document object, which 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.

When accessing a Fireworks document, consider the following points:

- To use a DOM function with a document other than the active document, use the following syntax; note that documentIndex is a zero-based index that specifies which document the command will affect.
  ```javascript
  fw.documents[documentIndex].functionName();
  ```
- To use a DOM function with the active document, use `fw.getDocumentDOM().functionName()` (for more information, see `fw.getDocumentDOM()`).

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:

```javascript
var expSetDir = fw.appExportSettingsDir;
```
**The Fireworks Object Model**

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 operations. 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 fragment has the same effect as the second and third code fragments. The `fw.getDocumentDOM()` function references the current document (for more information, see “Accessing a Fireworks document” on page 7).

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

**Formatting nonstandard data types**

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

- Some functions accept values that are Fireworks objects. For more information, see “The Fireworks Object Model” on page 7.
- Some functions accept a string in a specific format. Others accept 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 data type**

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

**Mask data type**

The format for a 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 the value of `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 the value of `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 data type
The format for a matrix is \{matrix: [float, float, float, float, float, float, float, float, float]\}. This guide 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 data type
The format for a point is \{x: float, y: float\}. For instance, `dom.addNewLine(startPoint, endPoint)` could look like the following example:

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

Rectangle data type
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:

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

Resolution data type
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:

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

Using Fireworks API functions

Three categories of API functions are described in this book: 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:

```javascript
fw.getDocumentDOM().deleteLayer(2);
```

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.

Working with selected elements
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 324). 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.

Using the common API
You can use the common Adobe API if you want commands to use a common syntax (and thus run a single command in multiple applications). 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().

app.setFloaterVisibility()
Identical to fw.setFloaterVisibility().

app.getRootDirectory()
Identical to the Fireworks object property appDir *

app.browseDocument()
Identical to fw.browseDocument().

Note: The app.getRootDirectory() function is useful if you want to use app.browseDocument() to view files within the application's folder.

Working with selected objects
When an object is selected, either programatically (for example, using the dom.selectAll() function) or by a user, you can return (get) or set the value of that object’s properties using common notation that will work on various objects. In other words, you can write a command that will get or set the value of an object’s properties whether the user selects a Text object, or an Image object, or any other recognized object. In Fireworks, a recognizable object is classified as one of the following element types:

- Hotspot
- SliceHotspot (basically, a slice)
- Path
- Group
- Instance
- Text
To test whether a text block is selected, type the following code:

```javascript
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 get values for the following read-only properties 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:

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

The following properties contain other read-only properties that you can return:

**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:

```javascript
effectName=fw.selection[0].effectList.effects[0].name;
```
Working with specific properties for selected elements

Some elements have specific properties that can be returned in addition to those that can be returned for any selected object (for more information, see “Working with properties for any selected object” on page 11). These specific properties are available for each of the following elements when the elements are selected.

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:

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

SliceHotspot

SliceHotspot is a subclass of Hotspot. A slice has all Hotspot 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:

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

Path

- pathAttributes

  *Note: For the complete list of path attributes properties, see “pathAttributes” on page 98.*

- randSeed
- textureOffset
- contours

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

```javascript
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 `pathAttributes` properties, see “pathAttributes” on page 98.

- `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 `pathAttributes` properties, see “pathAttributes” on page 98.

- `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 sub properties 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:

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

Global methods
The following are the global Fireworks methods, along with their argument data types and, where appropriate, acceptable values and notes.

alert(message)

Availability
Fireworks 3.

Usage
```
alert(message)
```

Arguments
- message A string containing the message to display.

Returns
- Nothing.

Description
- Displays the message in a modal alert box, along with an OK button.
confirm(message)

Availability
Fireworks 3.

Usage
confirm(message)

Arguments
message A string containing the message to display.

Returns
True if OK is clicked, false if Cancel is clicked.

Description
Displays a string in a modal alert box, along with OK and Cancel buttons.

prompt(caption, text)

Availability
Fireworks 3.

Usage
prompt(caption, text)

Arguments
caption A string containing the title of the dialog box.
text A string containing the prompt for the user.

Returns
The string entered if OK is clicked, null if Cancel is clicked.

Description
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.

write(arg1, arg2, ..., argN)

Availability
Fireworks 3.

Usage
write(arg1, arg2, ..., argN)
Arguments
arg1, arg2, ..., argN  Strings containing content for output.

Returns
An output file.

Description
Same as WRITE_HTML; WRITE_HTML was created to let you differentiate HTML output calls from other JavaScript calls in your code.

WRITE_CSS

Availability
Fireworks 3.

Usage
WRITE_CSS

Arguments
None.

Returns
An output file.

Description
Available only when exporting. Writes the CSS as an external file.

WRITE_HTML(arg1, arg2, ..., argN)

Availability
Fireworks 3.

Usage
WRITE_HTML(arg1, arg2, ..., argN)

Arguments
arg1, arg2, ..., argN  Strings containing content for output.

Returns
An output file.

Description
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 277.
fw.setUseAGMRenderingForSelection()

Availability
Fireworks 10.

Usage
fw.setUseAGMRenderingForSelection()

Arguments
A boolean value that defines whether AGMRendering is used or not.

Returns
Nothing.

Description
Sets the currently selected vectors to use the new AGM Rendering.

Example
fw.getDocumentDOM().setUseAGMRenderingForSelection(true);

Core objects

This section describes the set of core objects that are always available: Errors, Files, Find, and System. The Document object is described within its own chapter: see “The Document object” on page 28.

Note: For information on how to format nonstandard data types, such as rectangle or point, see “Formatting nonstandard data types” on page 8.

Errors object

All Errors object properties are read-only strings that are used to simplify the localizing of scripts. 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 list contains the properties of the Errors object alphabetically:

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

Files object

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

##### Method | Data type | Notes
--- | --- | ---
`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 the name of a file, which is expressed as `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 a value of `true` if the copy is successful; false otherwise.

`createDirectory(dirname)` | string | Creates the specified directory. Returns `true` if successful; false otherwise.

`createFile(fileURL, fileType, fileCreator)` | 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 `file://URL`. The last two arguments let you specify the file type and file creator strings. The `fileType` and `fileCreator` strings should each be strings of exactly four characters in length, for example:

```java
Files.createFile(newFile, "".txt", "FWMX");
```

`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.

`getDirectory(docname)` | string | Returns only the directory name from `docname`, which is expressed as `file://URL`. For example,

```java
Files.getDirectory("file:///work/logo.png") returns "file:///work".
```

`getExtension(docname)` | string | Returns the filename extension, if any, of `docname`. For example,

```java
Files.getExtension("birthday.png") returns ".png". If the filename has no extension, an empty string is returned. A filename that is expressed as `file://URL` is acceptable.
```

`getFilename(docname)` | string | Returns just the filename from `docname`, which is expressed as `file://URL`. For example,

```java
Files.getFilename("file:///work/logo.png") returns "logo.png".
```

`getLanguageDirectory()` | string | Returns the URL of the language directory associated with the currently running language.

`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.

---

Last updated 5/17/2012
## File Reference object

The File Reference object is used to refer to a specific open file. The following table lists the methods of the File Reference object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getTempFilePath({dirname})</code></td>
<td>string</td>
<td>The argument, if used, must be expressed as <code>file://URL</code>. 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 <code>dirname</code> is passed and is not null, the URL that is returned indicates a file in the specified directory rather than in the Temporary Files directory.</td>
</tr>
<tr>
<td><code>isDirectory(dirname)</code></td>
<td>string</td>
<td>The argument must be expressed as <code>file://URL</code>. Returns true if the specified URL refers to a directory that exists; false otherwise.</td>
</tr>
<tr>
<td><code>makePathFromDirAndFile(dirname, plainFilename)</code></td>
<td>string, string</td>
<td>The first argument must be expressed as <code>file://URL</code>. Concatenates the two arguments to return a file URL that references the specified filename in the specified directory. For example, <code>Files.makePathFromDirAndFile(&quot;file:///work/reports&quot;, &quot;logo.png&quot;)</code> returns <code>&quot;file:///work/reports/logo.png&quot;</code>.</td>
</tr>
<tr>
<td><code>open(docname, encoding, bWrite)</code></td>
<td>string, string, Boolean</td>
<td>The first argument must be expressed as <code>file://URL</code>. Opens the specified file for reading or writing. The second argument specifies the encoding. If the third argument is true, the file opens for writing; otherwise it opens for reading. If the file cannot be opened, returns null; otherwise, returns a File Reference object.</td>
</tr>
<tr>
<td><code>rename(docname, newPlainFilename)</code></td>
<td>string, string</td>
<td>The <code>docname</code> argument is a file path or a file URL to the file that you want to rename. The <code>newPlainFilename</code> argument is the new name to assign to the file. The <code>rename</code> method returns a URL path of the newly renamed file if successful; otherwise, <code>Fireworks</code> returns null.</td>
</tr>
<tr>
<td><code>setFilename(docname, newPlainFilename)</code></td>
<td>string, string</td>
<td>The first argument must be expressed as <code>file://URL</code>. Returns a file URL with <code>docname</code> replaced by <code>newPlainFilename</code>. For example, <code>Files.setFilename(&quot;file:///work/logo.png&quot;, &quot;oldlogo.png&quot;)</code> returns <code>&quot;file:///work/oldlogo.png&quot;</code>. 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 <code>rename()</code>.</td>
</tr>
<tr>
<td><code>swap(docname1, docname2)</code></td>
<td>string, string</td>
<td>Each argument must be expressed as a <code>file://URL</code>. 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.</td>
</tr>
</tbody>
</table>
Find object
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 (•).

### Finding and replacing text

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>whatToFind</td>
<td>string</td>
<td>In the format: &quot;text&quot;</td>
</tr>
<tr>
<td>find</td>
<td>string</td>
<td>Text to find.</td>
</tr>
<tr>
<td>matchCase</td>
<td>Boolean</td>
<td>If set to true, the search is case-sensitive. Defaults to false.</td>
</tr>
<tr>
<td>numItemsReplaced•</td>
<td>Integer</td>
<td>Number of replacements done in a file using Find and Replace.</td>
</tr>
<tr>
<td>regExp</td>
<td>Boolean</td>
<td>If set to true, the find and replace text is interpreted as a regular expression. The default is false.</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>Text to use as replacement text.</td>
</tr>
<tr>
<td>wholeWord</td>
<td>Boolean</td>
<td>If set to true, only whole words matching the search text are found. The default is false.</td>
</tr>
</tbody>
</table>

### Finding and replacing fonts and styles

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>whatToFind</td>
<td>string</td>
<td>In the format: &quot;font&quot;</td>
</tr>
<tr>
<td>find</td>
<td>string</td>
<td>Name of font to find.</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>Name of font to use as replacement.</td>
</tr>
</tbody>
</table>
## Finding and replacing colors, fills, strokes, and effects

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>whatToFind</td>
<td>string</td>
<td>In the format: &quot;color&quot;</td>
</tr>
<tr>
<td>find</td>
<td>string</td>
<td>A color string that specifies the color to find (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>A color string that specifies the color to use as a replacement (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>fills</td>
<td>Boolean</td>
<td>If set to true, fills that match the specified colors are replaced.</td>
</tr>
<tr>
<td>strokes</td>
<td>Boolean</td>
<td>If set to true, strokes that match the specified colors are replaced.</td>
</tr>
<tr>
<td>effects</td>
<td>Boolean</td>
<td>If set to true, effects that match the specified colors are replaced.</td>
</tr>
</tbody>
</table>

## Finding and replacing URLs

<table>
<thead>
<tr>
<th>Property</th>
<th>Data types</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>whatToFind</td>
<td>string</td>
<td>In the format: &quot;url&quot;</td>
</tr>
<tr>
<td>find</td>
<td>string</td>
<td>URL to find, which is expressed as file://URL.</td>
</tr>
<tr>
<td>replace</td>
<td>string</td>
<td>URL to use as replacement text, which is expressed as file://URL.</td>
</tr>
<tr>
<td>wholeWord</td>
<td>Boolean</td>
<td>If set to true, only whole words that match the search text are found. The default is false.</td>
</tr>
<tr>
<td>matchCase</td>
<td>Boolean</td>
<td>If set to true, the search is case sensitive. Defaults to false.</td>
</tr>
<tr>
<td>regexp</td>
<td>Boolean</td>
<td>If set to true, the find and replace text is interpreted as a regular expression. The default value is false.</td>
</tr>
</tbody>
</table>
Finding and replacing non-websafe colors with the closest websafe color

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>whatToFind</td>
<td>string</td>
<td>In the format: &quot;nonwebcolor&quot;</td>
</tr>
<tr>
<td>effects</td>
<td>Boolean</td>
<td>If set to true, colors in effects are replaced. The default value is false.</td>
</tr>
<tr>
<td>fills</td>
<td>Boolean</td>
<td>If set to true, colors in fills are replaced. The default value is false.</td>
</tr>
<tr>
<td>strokes</td>
<td>Boolean</td>
<td>If set to true, colors in strokes are replaced. The default value is false.</td>
</tr>
</tbody>
</table>

System object

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

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>osName</td>
<td>string</td>
<td>Returns the name of the operating system under which Fireworks is running.</td>
</tr>
<tr>
<td>controlFaceColor</td>
<td>string</td>
<td>Returns the system color used for the control and panel faces (Windows-only property).</td>
</tr>
<tr>
<td>controlHilightColor</td>
<td>string</td>
<td>Returns the system color used for control highlights (Windows-only property).</td>
</tr>
<tr>
<td>controlShadowColor</td>
<td>string</td>
<td>Returns the system color used for control shadows (Windows-only property).</td>
</tr>
<tr>
<td>controlDarkShadowColor</td>
<td>string</td>
<td>Returns the system color used for control dark shadows (Windows-only property).</td>
</tr>
<tr>
<td>hilightItemColor</td>
<td>string</td>
<td>Returns the system color used for highlighting selections (Windows-only property).</td>
</tr>
<tr>
<td>hilightTextColor</td>
<td>string</td>
<td>Returns the system color used for highlighting selected text (Windows-only property).</td>
</tr>
<tr>
<td>textColor</td>
<td>string</td>
<td>Returns the system color used for text (Windows-only property).</td>
</tr>
<tr>
<td>menuColor</td>
<td>string</td>
<td>Returns the system color used for menu backgrounds (Windows-only property).</td>
</tr>
<tr>
<td>menuTextColor</td>
<td>string</td>
<td>Returns the system color used for text in menus (Windows-only property).</td>
</tr>
</tbody>
</table>

Adobe AIR Extension

AIRext.PreviewDocument()

Availability
Fireworks 10.
Usage
status= AIRextPreviewDocument(appXMLPath, siteRootPath);

Arguments
appXMLPath Path to the application descriptor.
siteRootPath Path to the site root folder in which the file that need to be previewed reside.

Returns
0 if the operation succeeded.

Description
Calls the ADL tool to preview an HTML document as an Adobe AIR application. The path to file that needs to be previewed is not specified because it is included in the application descriptor; this path is relative to the site root.

AIRext.PackageApplication()

Availability
Fireworks 10.

Usage
status = AIRextPackageApplication(siteFullPath, packagePath, fileList);

Arguments
siteFullPath Path to the site root folder in which all files that needs to be included in package reside.
packagePath Full path of the resulting Adobe AIR package.
fileList Array of paths to files and folders that need to be included in package; all paths need to be relative to the site root.

Returns
0 if the operation succeeded; 1 if an error occurred; 2 if the ADT tool generated an error (this error is saved into an error log file).

Description
Calls the ADT tool package an Adobe AIR application.

AIRext.CreatePackage()

Availability
Fireworks 10.

Usage
status = AIRextCreatePackage(siteFullPath, packagePath, fileList, appXMLPath, certificatePath, password);
Arguments

`siteFullPath` Path to the site root folder in which all files that need to be included in package reside.

`packagePath` Full path of the resulting Adobe AIR package.

`fileList` Array of paths to files and folders that need to be included in package; all paths need to be relative to the site root.

`appXMLPath` Path to the application descriptor (if blank or missing, defaults to application.xml in site root).

`certificatePath` Path to the digital certificate with which to sign the package (if blank or missing, an unsigned AIR file will be created instead of a signed AIR file)

`password` Password for the specified certificate (if blank or missing, user will be prompted for the password).

Returns

0 if the operation succeeded.

Description

Calls the ADT tool to create a signed AIR package.

**AIRext.CheckCertificatePassword()**

Availability

Fireworks 10.

Usage

```javascript
status = AIRext.CheckCertificatePassword(certificatePath, password);
```

Arguments

`certificatePath` Path to the digital certificate with which to sign the package.

`password` Password for the specified certificate.

Returns

Exitcode from ADT; 0 if OK, 7 if could not access certificate, and so on.

Description

Calls the ADT tool to check a certificate password.

**AIRext.SaveDigSigPassword()**

Availability

Fireworks 10.

Usage

```javascript
AIRext.SaveDigSigPassword(certificatePath, password);
```

Arguments

`certificatePath` Path to the digital certificate whose password is being saved.
password: Password for the specified certificate.

Returns
0 if saved successfully.

Description
Saves the password for the given digital certificate, for the current session of Fireworks (no persistent storage).

AIRext.GetDigSigPassword()

Availability
Fireworks 10.

Usage
configurationPath = AIRext.GetDigSigPassword();

Arguments
None.

Returns
The previously saved password for the given digital certificate, or an empty string.

Description
Returns the name of the generated password file.

AIRext.GetLastErrorLogName()

Availability
Fireworks 10.

Usage
errLogName = AIRext.GetLastErrorLogName();

Arguments
None.

Returns
The name of the error log file if it was created, or an empty string.

Description
Returns the name of the generated error log file if one was created.
AIRext.GetAIRInstallPath()

Availability
Fireworks 10.

Usage
configurationPath = AIRext.GetAIRInstallPath();

Arguments
None.

Returns
The full system path to the current user configuration folder (for example, "C:\Documents and settings\...\Configuration").

Description
Returns the full system path name and path of the generated file.

AIRext.GetJREVersion()

Availability
Fireworks 10.

Usage
jreVersObj = AIRext.GetJREVersion();

Arguments
None.

Returns
An object with two properties - maxVersion and minVersion; if an error occurs the values of the maxVersion and minVersion will be -1.

Description
Get the minimum and maximum versions of the JRE.

AIRext.CreateCertificate()

Availability
Fireworks 10.

Usage
status = AIRext.CreateCertificate(certificatePath, password, keyType, publisher, orgName, orgUnit, country);
Arguments

certificatePath  Path to digital certificate file that will be created (required).

password  Password for the created certificate (required)

keyType  Encryption key type (either "1024-RSA" or "2048-RSA" (required).

publisher  Name of publisher (required).

orgName  Name of organization (optional).

orgUnit  Organizational unit (optional).

country  2-letter code for country (optional, and length is not enforced).

Returns

0 if operation succeeded.

Description

Calls the ADT to create a self-signed digital certificate.
# Chapter 3: The Document object

This chapter describes the Fireworks Document object and functions.

## Document object properties

The following table lists the properties of the Document object, along with their data types, 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 “Property inspector functions” on page 321.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>backgroundColor</td>
<td>string</td>
<td>A color string that specifies the document canvas color (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>backgroundURL</td>
<td>string</td>
<td>Sets a general URL for a document that uses a hotspot. Everything that is not covered by the hotspot has the background URL.</td>
</tr>
<tr>
<td>brushes •</td>
<td>array</td>
<td>Array of Brush objects that are available for use in the document (for more information, see “Brush object” on page 243).</td>
</tr>
<tr>
<td>currentFrameNum</td>
<td>zero-based index</td>
<td>The index of the current frame.</td>
</tr>
<tr>
<td>currentLayerNum</td>
<td>zero-based index</td>
<td>The index of the current layer.</td>
</tr>
<tr>
<td>defaultAltText</td>
<td>string</td>
<td>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 the Document-Specific tab of the HTML Setup dialog box.</td>
</tr>
<tr>
<td>docTitleWithoutExtension</td>
<td>string</td>
<td>The title of the document file, without any file extension. If the document has not been saved, this string is empty.</td>
</tr>
<tr>
<td>exportFormatOptions</td>
<td>object</td>
<td>Identical to exportOptions. Included for backward compatibility with Fireworks 2.</td>
</tr>
<tr>
<td>exportOptions</td>
<td>object</td>
<td>ExportOptions object (for more information, see “ExportOptions object” on page 260).</td>
</tr>
<tr>
<td>exportSettings</td>
<td>object</td>
<td>ExportSettings object (for more information, see “ExportSettings object” on page 263).</td>
</tr>
<tr>
<td>filePathForRevert</td>
<td>string</td>
<td>The path to the file from which this document was opened, which is expressed as file://URL, or null if created from scratch.</td>
</tr>
<tr>
<td>filePathForSave</td>
<td>string</td>
<td>The location to which this document was saved, which is expressed as file://URL, or null if never saved.</td>
</tr>
<tr>
<td>fills •</td>
<td>array</td>
<td>Array of Fill objects that are available for use in the document (for more information, see “Fill object” on page 265).</td>
</tr>
<tr>
<td>frameCount</td>
<td>integer</td>
<td>The number of frames in the current document.</td>
</tr>
<tr>
<td>frameLoopingCount</td>
<td>integer</td>
<td>−1 — don’t repeat 0 — repeat forever &gt; 0 — repeat this number of times</td>
</tr>
</tbody>
</table>
### Property | Data type | Notes
---|---|---
frames | array | Array of Frame objects in the document (for more information, see "Frame object" on page 266).
gammaPreview | Boolean | If set to `true`, the document should be previewed in opposite-platform gamma. If set to `false`, the document colors are unadjusted.
gradients | array | Array of Gradient objects that are available for use in the document (for more information, see "Gradient object" on page 267).
ggridColor | string | A color string that specifies the color of the grid display (for more information, see "Color string data type" on page 8).
ggridOrigin | point | Used to set the origin of the grid. Corresponds to the point set when dragging the ruler origin from the upper-left of the document when rulers are visible.
ggridSize | point | `gridSize.x` is the horizontal grid size; `gridSize.y` is the vertical grid size.
guides | object | Guides object (for more information, see "Guides object" on page 267).
height | integer | Total height of the document, in pixels. To find the bottom edge of the document, use `document.top + document.height`.
isDirty | Boolean | Set to `true` if the document was modified since the last time it was saved.
isPaintMode | Boolean | Set to `true` if the document is currently in paint-mode editing, `false` otherwise.
isSymbolDocument | Boolean | Set to `true` if the document is a Symbol or Button document, `false` if it is an ordinary document. You might see this when looking through the list of open documents and one is a symbol-editing window.
isValid | Boolean | Set to `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 eventuality.)
lastExportDirectory | string | The path to the last directory to which the file was exported, which is expressed as `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 (for more information, see "Layer object" on page 268).
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`.
## Property | Data type | Notes
--- | --- | ---
mapType | string | Acceptable values are "client", "server", and "both". Corresponds to the image-map type selected in File > HTML Setup.
matteColor | string | A color string that corresponds to the matte color specified in the Optimize panel (for more information, see “Color string data type” on page 8). This string is used by the useMatteColor property.
onionSkinAfter | integer | The number of frames after the current frame to show through 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.
onionSkinBefore | integer | Similar to the onionSkinAfter property, but refers to the number of frames to show through onion skinning before the current frame.
pagesCount | integer | Returns the number of pages in the current document.
pageName | string | Returns the page name of the current page.
patterns | object | The list of internal pattern names.
pathAttributes | object | The PathAttrs object (for more information, see "PathAttrs object" on page 268). 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 “The pngText property” on page 31.
resolution | float | Document resolution, in pixels per unit (for more information, see resolutionUnits). The range is 1 to 5000.
resolutionUnits | string | Units to be used with the resolution property. Acceptable values are "inch" and "cm".
savedSelections | object | Array of the saved bitmap selections in the active document.
textures | array | Array of Texture objects that are available for use in the document (for more information, see "Texture object" on page 259).
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 set to true, the matteColor property is used when exporting documents with transparent backgrounds. If set to 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.
currentPageNum | integer | The index of the current page.
topLayers | array | Array of top level layers within the document objects.
The **pngText property**

Fireworks maintains the following fields for use with the pngText property:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreationTime</td>
<td>The date and time the document was created.</td>
</tr>
<tr>
<td>Software</td>
<td>The software used to create the document. The current version of Fireworks always sets this value to &quot;Adobe Fireworks CS5.&quot;</td>
</tr>
</tbody>
</table>

You can edit these fields 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.

**Document functions**

As discussed in an earlier section, you get and set document properties by calling functions as methods of the document’s Document Object Model (DOM) (see “Accessing a Fireworks document” on page 7). 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:

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

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

**dom.addBehavior()**

**Availability**

Fireworks 3.

**Usage**

dom.addBehavior(action, event, eventIndex)

**Arguments**

- **action** 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 dom.addBehavior() function” on page 32.

- **event** The event that triggers the behavior. Acceptable values are “onMouseOver”, “onMouseOut”, “onLoad”, and “onClick”.

- **eventIndex** An integer value that specifies the location where the behavior should be added, starting with 0 (although, to specify the end location, pass –1 here).

**Returns**

Nothing.
**Description**  
Adds a specified behavior to the selected hotspots and slices.

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

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

**See also**  
“dom.removeBehavior()” on page 117

**Using the dom.addBehavior() function**  
The following code shows the syntax for `dom.addBehavior()`:

```javascript
fw.getDocumentDOM().addBehavior(action, event, eventindex);
```

The first argument is a string that specifies the behavior to be added; see `dom.addBehavior()`. The information in this section describes the acceptable values for the `action` argument that is passed to `dom.addBehavior()`.

**MM_nbGroup [down]**

**Availability**  
Fireworks 3.

**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 an integer value that specifies the frame to swap, starting with 0 (although, 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).

**Description**  
Sets a navigation bar Down behavior.

**Example**

```javascript
fw.getDocumentDOM().addBehavior("MM_nbGroup('down','navbar1','-1,2,"",1)", "onClick", -1);
```

**MM_nbGroup [highlight]**

**Availability**  
Fireworks 3.
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 an integer value that specifies the frame to swap, starting with 0 (although, 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 an integer value that specifies the frame to use as a highlight image, starting with 0 (although, 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.

Description

Sets a navigation bar highlight behavior.

Example

```javascript
fw.getDocumentDOM().addBehavior("MM_nbGroup('over',-1,1,"",1,0,3,"")", "onMouseOver", -1);
```

**MM_nbGroup [image]**

Availability

Fireworks 3.

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).

Description

Sets a navigation bar image behavior.

Example

```javascript
fw.getDocumentDOM().addBehavior("MM_nbGroup('all',1,0)", "onMouseOver", -1);
```
**MM_nbGroup [out]**

**Availability**
Fireworks 3.

**Arguments**
- `type` Pass "out" for `type`.

**Description**
Sets a navigation bar restore behavior.

**Example**
```
fw.getDocumentDOM().addBehavior("MM_nbGroup\('out'\)", "onMouseOut", -1);
```

**MM_simpleRollover**

**Availability**
Fireworks 3.

**Arguments**
None.

**Description**
Adds a simple rollover behavior.

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

**MM_statusMessage**

**Availability**
Fireworks 3.

**Arguments**
- `message` A string that specifies the status message to appear.

**Description**
Sets a status bar message.

**Example**
```
fw.getDocumentDOM().addBehavior("MM_statusMessage("Status Message!"), "onMouseOver", -1);
```

**MM_swapImage**

**Availability**
Fireworks 3.
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 an integer value that specifies the frame to swap, starting with 0 (although, 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).

Description
Adds a swap image behavior.

Example
fw.getDocumentDOM().addBehavior("MM_swapImage(-1,1,"",1,1)", "onMouseOver", -1);

MM_swapImgRestore

Availability
Fireworks 3.

Arguments
None.

Example
fw.getDocumentDOM().addBehavior("MM_swapImgRestore()", "onMouseOut", -1);

Description
Adds a swap image restore behavior.

dom.addElementMask()

Availability
Fireworks 4.

Usage
dom.addElementMask(mode, {bEnterMaskEditMode})

Arguments
mode 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.

bEnterMaskEditMode If bEnterMaskEditMode (optional) is set to true, Fireworks enters mask-edit mode on the newly added mask; if omitted, it defaults to false.
Returns
Nothing.

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.

**dom.addFrames()**

Availability
Fireworks 3, enhanced in Fireworks 4.

Usage
```javascript
dom.addFrames(howMany, where, {bAdvanceActiveFrame})
```

Arguments
- **howMany**  
  An integer that specifies how many frames to add.
- **where**  
  The location where frames should be added. Acceptable values for where are "beginning", "before current", "after current", and "end".
- **bAdvanceActiveFrame**  
  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.

Description
Adds one or more frames to the document.

Example
The following command adds one frame after the current frame but does not change the active frame:
```javascript
fw.getDocumentDOM().addFrames(1, "after current", false);
```

**dom.addGuide()**

Availability
Fireworks 3.

Usage
```javascript
dom.addGuide(position, guidekind)
```

Arguments
- **position**  
  A floating-point value that specifies the x or y coordinate at which to add the guide.
 guidekind  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.

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

Example
The following command adds a vertical guide at the x coordinate of 217:

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

dom.addMasterPageLayer()

Availability
Fireworks CS3.

Usage
dom.addMasterPageLayer()

Arguments
To add a master page layer, the document must already have a master page in it. Include the level where the master page layer should be inserted. For example:

fw.getDocumentDOM().addMasterPageLayer(-1)

Returns
Nothing.

Description
Adds a master page layer to the bottom of the layer hierarchy for the current page.

dom.addNewHotspot()

Availability
Fireworks 3.

Usage
dom.addNewHotspot(hotspot-kind, hotspot-shape, boundingRectangle)

Arguments
hotspot-kind  Acceptable values are "hotspot" and "slice".

hotspot-shape  Acceptable values are "rectangle" and "oval".

boundingRectangle  A rectangle that specifies the bounds within which the hotspot is placed (see "Rectangle data type" on page 9).
Returns
Nothing.

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

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

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

**dom.addNewImage()**

Availability
Fireworks 3.

Usage
dom.AddNewImage(boundRectangle, bEnterPaintMode)

Arguments
boundRectangle A rectangle that specifies the bounds of the image to be added (see “Rectangle data type” on page 9). 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.

bEnterPaintMode If bEnterPaintMode is true, the application immediately enters bitmap mode for the new image.

Returns
Nothing.

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

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

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

**dom.addNewImageViaCopy()**

Availability
Fireworks MX.

Usage
dom.addNewImageViaCopy()

Arguments
None.
Returns
Nothing.

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.

dom.addNewImageViaCut()

Availability
Fireworks MX.

Usage
dom.addNewImageViaCut()

Arguments
None.

Returns
Nothing.

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.

dom.addNewLayer()

Availability
Fireworks 3.

Usage
dom.addNewLayer(name, bshared)

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

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

Description
Adds a new layer to the document and makes it the current layer.
EXTENDING FIREWORKS
The Document object

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

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

**dom.addNewLine()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.addNewLine(startPoint, endPoint)
```

**Arguments**
- `startPoint` and `endPoint` Points that specify the x,y coordinates between which the path is added (see “Point data type” on page 9).

**Returns**
Nothing.

**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.

**Example**
The following command adds a new line between the specified coordinates:

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

**dom.addNewOval()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.addNewOval(boundingRectangle)
```

**Arguments**
- `boundingRectangle` A rectangle that specifies the bounds of the oval to be added (see “Rectangle data type” on page 9).

**Returns**
Nothing.

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

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

**dom.addNewPage()**

**Availability**
Fireworks CS3.

**Usage**
```javascript
dom.addNewPage()
```

**Arguments**
None

**Returns**
Nothing.

**Description**
Adds a new page to the current document.

**dom.addNewRectangle()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.addNewRectangle(boundingRectangle, roundness)
```

**Arguments**
- `boundingRectangle` A rectangle that specifies the bounds within which the new rectangle is added (see “Rectangle data type” on page 9).
- `roundness` A floating-point value between 0 and 1 that specifies the roundness to use for the corners (0 is no roundness, 1 is 100% roundness).

**Returns**
Nothing.

**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.

**Example**
The following command adds a new rectangle with no round corners within the specified coordinates:

```javascript
fw.getDocumentDOM().addNewRectangle({left:0, top:0, right:100, bottom:100}, 0);
```
See also
dom.addNewRectanglePrimitive()

dom.addNewRectanglePrimitive()

Availability
Fireworks 4.

Usage
dom.addNewRectanglePrimitive(boundingRectangle, roundness)

Arguments
boundingRectangle A rectangle that specifies the bounds within which the new rectangle primitive is added (see “Rectangle data type” on page 9).

roundness A floating-point value between 0 and 1 that specifies the “roundness” to use for the corners (0 is no roundness, and 1 is 100% roundness).

Returns
Nothing.

Description
Adds a new rectangle primitive that fits in 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 preserves its rectangular quality; that is, if you drag a corner, it remains a rectangle rather than becoming a quadrilateral.

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);

See also
dom.addNewRectangle(), fw.ungroupPrimitives()

dom.addNewSinglePointPath()

Availability
Fireworks 3.

Usage
dom.addNewSinglePointPath(controlPointFirst, mainpoint, controlPointLast, bCopyAttrs)
Arguments
controlPointFirst, mainpoint, and controlPointLast
Points that specify the x,y coordinates of the preceding control point, the main point, and the following control point of the Bezier path (see “Point data type” on page 9).

bCopyAttrs
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.

Description
Adds a new path that consists of a single Bezier 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.

Example
The following command adds a new path that consists of a single Bezier point at the specified coordinates and copies the path’s stroke and fill from the document’s current stroke and fill settings:

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

dom.addNewStar()

Availability
Fireworks 3

Usage
dom.AddNewStar(numSides, spikiness, bIsStar, centerPoint, outsidePoint)

Arguments
numSides
An integer that specifies the number of sides of the new path.

spikiness
A floating-point 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.

bIsStar
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 data type” on page 9).

outsidePoint
Specifies a point on the radius of the star or polygon.

Returns
Nothing.

Description
Adds a new star- or polygon-shaped path.

Example
The following command adds a five-sided star:
EXTENDING FIREWORKS

The Document object

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

dom.addNewSubLayer()

Availability
Fireworks CS3.

Usage
`dom.addNewSubLayer(index, name, shared)`

Arguments
- `index`: A long value that specifies the index of the parent layer for the new sub layer.
- `name`: A string that specifies the name for the new sub layer. If `name` is `null`, a new layer name is generated.
- `shared`: A Boolean value that specifies whether the new sub layer is shared.

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

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

Example
The following command adds a new unshared sub layer to layer index 0 with a default name that is generated by Fireworks:

```javascript
fw.getDocumentDOM().addNewSubLayer(0, null, false);
```

dom.addNewSymbol()

Availability
Fireworks 3.

Usage
`dom.addNewSymbol(type, name, bAddToDoc, status)`

Arguments
- `type`: Acceptable values are "graphic", "button", or "animation".
- `name`: A string that specifies the name of the symbol.
- `bAddToDoc`: A Boolean value. 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.
- `status`: A Boolean value that toggles 9-slice scaling between enabled and disabled.

Returns
Nothing.
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.

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

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

**dom.addNewText()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.addNewText(boundingBox, bInitFromPrefs)
```

**Arguments**

*boundingBox* A rectangle that specifies the bounds within which to place the new text box (see "Rectangle data type" on page 9).

*bInitFromPrefs* 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.

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

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

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

**dom.addSwapImageBehaviorFromPoint()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.addSwapImageBehaviorFromPoint(where)
```

**Arguments**

*where* A point that specifies the x,y coordinates of the hotspot or slice that contains the swap image behavior to be added (see “Point data type” on page 9).
Returns
true if the swap image behavior was added; false if no suitable hotspot was at the specified location.

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.

dom.adjustExportToSize()

Availability
Fireworks 3.

Usage
dom.AdjustExportToSize(sizeInBytes, bOkToIncreaseSize)

Arguments
sizeInBytes  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.

Description
Adjusts the export settings as specified.

dom.adjustFontSize()

Availability
Fireworks MX.

Usage
dom.adjustFontSize(amount)

Arguments
amount  The amount, specified in points, by which to change the font size. Positive values (such as "2pt") increase the size, while negative values (such as "-1pt") decrease the size.

Returns
Nothing.
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.

**dom.align()**

**Availability**
Fireworks 3. Align to canvas parameter is only available in Fireworks 8.

**Usage**

```javascript
dom.align(alignmode, alignToCanvas)
```

**Arguments**

- **alignmode** Acceptable values are "left", "right", "top", "bottom", "center vertical", and "center horizontal".
- **alignToCanvas** Boolean. Determines if the alignment is to the canvas or items. The default value is false.

**Returns**

Nothing.

**Description**

Aligns the selection.

**dom.appendPointToHotspot()**

**Availability**

Fireworks 3.

**Usage**

```javascript
dom.appendPointToHotspot(pt, tolerance)
```

**Arguments**

- **pt** A point that specifies the x,y coordinates of the point to be added (see “Point data type” on page 9).
- **tolerance** A floating-point 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.

**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.
**dom.appendPointToPath()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.appendPointToPath(contourIndex, ptToInsertBefore, controlPointFirst, mainPoint, controlPointLast)
```

**Arguments**
- **contourIndex** An zero-based index value that specifies the contour to which the Bezier point is appended. For paths with multiple contours, the contours are in an arbitrary order.
- **ptToInsertBefore** A zero-based index value 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** 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 data type” on page 9).

**Returns**
Nothing.

**Description**
Appends a Bézier point to the selected path.

**See also**
`dom.insertPointInPath()`

**dom.appendPointToSlice()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.appendPointToSlice(pt, tolerance)
```

**Arguments**
- **pt** A point that specifies the x,y coordinates of the point to be added (see “Point data type” on page 9).
- **tolerance** A floating-point value $\geq 0$ 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.

**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.
**dom.applyCharacterMarkup()**

**Availability**
Fireworks 3, updated in Fireworks 4.

**Usage**
`dom.applyCharacterMarkup(tag)`

**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.

**Description**
Applies the specified character markup to the selected text.

**dom.applyCurrentFill()**

**Availability**
Fireworks 3.

**Usage**
`dom.applyCurrentFill(bNoNullFills)`

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

**Returns**
Nothing.

**Description**
Applies the document’s current fill to the selection.

**Example**
The following command applies the current fill to the selection:

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

**dom.applyEffects()**

**Availability**
Fireworks 3.
Usage

\texttt{dom.ApplyEffects\{\texttt{effectList}\}}

Arguments

define \texttt{effectList} An EffectList object (see "EffectList object" on page 254). If \texttt{effectList} is null, this function removes all effects from the selection.

Returns

Nothing.

Description

Applies the specified effects to the selection.

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 object" on page 251):

\begin{verbatim}
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" });
\end{verbatim}

\texttt{dom.applyFontMarkup()}

Availability

Fireworks 3.

Usage

\texttt{dom.applyFontMarkup\{\texttt{fontAttribute, value}\}}

Arguments

define \texttt{fontAttribute} Acceptable values for \texttt{fontAttribute} are "size" and "face".

\texttt{value} If \texttt{fontAttribute} is "size", \texttt{value} must be of the form "XXXpt" to specify a point size; a simple numeric value is not allowed.

Returns

Nothing.

Description

Applies the specified font markup attribute to the selected text.

\texttt{dom.applyStyle()}

Availability

Fireworks 3.
Usage

`dom.applyStyle(styleName, styleIndex)`

**Arguments**

- **styleName**: A string that specifies the style name to be applied. The style group from which the style is being applied should be selected.
- **styleIndex**: An index to the style to apply. This 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.

**Description**

Applies the specified style to the selection.

**Example**

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

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

**dom.arrange()**

**Availability**

Fireworks 3.

**Usage**

`dom.arrange(arrangemode)`

**Arguments**

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

**Returns**

Nothing.

**Description**

Arranges the selection.

**Example**

The following command brings the selected items to the front:

```javascript
fw.getDocumentDOM().arrange("front");
```
**dom.attachTextInPath()**

**Availability**
Fireworks CS5.

**Usage**
dom.attachTextInPath()

**Arguments**
None.

**Returns**
Nothing.

**Description**
Attaches the selected text inside the selected path. If no text and path are selected, no action occurs.

**Example**
When two items are selected (one a text block and the other a shape), the following command attaches the text block inside the shape's path:

fw.getDocumentDOM().attachTextInPath();

**dom.attachTextToPath()**

**Availability**
Fireworks 3.

**Usage**
dom.attachTextToPath()

**Arguments**
None.

**Returns**
Nothing.

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

**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.breakLink()**

**Availability**
Fireworks 3.

**Usage**
dom.breakLink()

**Arguments**
None.

**Returns**
Nothing.

**Description**
Breaks the link between the symbol and the instance.

**dom.changeCurrentPage()**

**Availability**
Fireworks CS3.

**Usage**
dom.changeCurrentPage(pageNum)

**Arguments**

- **pageNum**: An long value that specifies the page number of the page that will become the active page.

**Returns**
Nothing.

**Description**
Changes the currently active page to the specified page number. The page number is in an array and the first page is numbered 0.

**dom.changeGuide()**

**Availability**
Fireworks 3.

**Usage**
dom.changeGuide(currentPosition, newPosition, guidekind)

**Arguments**

- **currentPosition**: A floating-point value that specifies the current position of the guide.
newPosition A floating-point value that specifies the new position of the guide.

guidekind 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.

Description Moves a guide’s position to a new location.

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

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

dom.changeNineScaleGuide()

Availability Fireworks CS3.

Usage

```
dom.changeNineScaleGuide(oldpos, newpos, guidekind)
```

Arguments

oldpos A double precision value that specifies the current position of the guide.

newpos A double precision value that specifies the new position of the guide.

guidekind 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.

Description Moves a 9-slice scaling guide’s position to a new location.

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

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

dom.changeSliceGuide()

Availability Fireworks MX.
Usage

dom.changeSliceGuide(currentPosition, newPosition, guidekind, isMagneticDrag, isSingleDrag)

Arguments

currentPosition A floating-point value that specifies the current position of the slice guide to be moved.

newPosition A floating-point value that specifies the new position of the slice guide.

guidekind Acceptable values are "horizontal" and "vertical". If the value of guidekind is "horizontal", Fireworks assumes that the specified positions are y coordinates; if "vertical", the specified positions are x coordinates.

isMagneticDrag 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.

isSingleDrag A Boolean value that determines whether the operation is performed only on the selected slice or on all slices that are affected by the slice guide. If isSingleDrag is true, Fireworks performs only the changeSliceGuide() action on the selected slice.

Returns

Nothing.

Description

Moves a slice guide’s position to a new location, which resizes any rectangular slices that abut the guide. An argument 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.

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.

Usage

dom.clearJPEGMask()

Arguments

None.

Returns

Nothing.
Description
Clears the "Selective JPEG mask" for the document.

**dom.clipCopy()**

*Availability*
Fireworks 3.

*Usage*
```javascript
dom.clipCopy()
```

*Arguments*
None.

*Returns*
Nothing.

*Description*
Copies the selection to the clipboard.

*Example*
The following command copies the selected items to the clipboard:
```javascript
fw.getDocumentDOM().clipCopy();
```

**dom.clipCopyAsPaths()**

*Availability*
Fireworks MX.

*Usage*
```javascript
dom.clipCopyAsPaths()
```

*Arguments*
None.

*Returns*
Nothing.

*Description*
Copies the selection to the clipboard in Adobe Illustrator format.

*Example*
The following command copies the selected items to the clipboard in Adobe Illustrator format:
```javascript
fw.getDocumentDOM().clipCopyAsPaths();
```
The Document object

dom.clipCopyFormats()

Availability
Fireworks MX.

Usage
dom.clipCopyFormats(format)

Arguments
format  The graphics format for the selection. For example, "AICB" is the Adobe Illustrator format.

Returns
Nothing.

Description
Copies the selection to the clipboard using the specified format.

dom.clipCopyJsToExecute()

Availability
Fireworks 9.

Usage
dom.clipCopyJsToExecute(string)

Arguments
string  The JavaScript code that is copied on to the clipboard that you want to execute.

Returns
Nothing.

Description
Copies the JavaScript command on to the clipboard.

dom.clipCut()

Availability
Fireworks 3.

Usage
dom.clipCut()

Arguments
None.
Returns
Nothing.

Description
Cuts the selection to the clipboard.

Example
The following command cuts the selected items and places them on the clipboard:

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

**dom.clipPaste()**

Availability
Fireworks 3, updated in Fireworks 4.

Usage

```javascript
dom.clipPaste({whatIfResolutionDifferent}, {whatIfPastingIntoElementMask})
```

Arguments

- **whatIfResolutionDifferent** 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** An optional argument, added in Fireworks 4, that 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 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.

Description
Pastes the clipboard contents into the document.

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.

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

**dom.clipPasteAsMask()**

Availability
Fireworks 4.
Usage

dom.clipPasteAsMask(whatIfResolutionDifferent, masktype, maskReplaceOptions)

Arguments

whatIfResolutionDifferent  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".

maskReplaceOptions  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.

Description

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

dom.clipPasteAttributes()

Availability

Fireworks 3.

Usage

dom.clipPasteAttributes()

Arguments

None.

Returns

Nothing.

Description

Pastes the attributes from the clipboard onto the selection.

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.

**Usage**
dom.clipPasteFromChannelToChannel(fromChannel, toChannel)

**Arguments**

- **fromChannel** 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.

- **toChannel** If the currently selected element is a bitmap, the toChannel argument is used to specify where to paste the color data.

**Returns**
Nothing.

**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.

**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()).

**Usage**
dom.clipPasteInside({whatIfResolutionDifferent})

**Arguments**

- **whatIfResolutionDifferent** 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.
Description
Pastes the clipboard contents into the selection, and places the selected element into the element mask for the pasted elements. 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.

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

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

### dom.cloneSelection()

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.cloneSelection()
```

**Arguments**
None.

**Returns**
Nothing.

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

**Example**
The following command copies the selected items on top of the original items:

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

**See also**
- `dom.duplicateSelection()`

### dom.close()

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.close(bPromptToSaveChanges)
```
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.

Returns
Nothing.

Description
Closes the document.

dom.convertAnimSymbolToGraphicSymbol()

Availability
Fireworks 4.

Usage
dom.convertAnimSymbolToGraphicSymbol()

Arguments
None.

Returns
Nothing.

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

See also
dom.convertToAnimSymbol(), dom.convertToSymbol()

dom.convertMarqueeToPath()

Availability
Fireworks 7.

Usage
dom.convertMarqueeToPath()

Arguments
None.

Returns
Nothing.
Description
Converts marquee selection to path.

`dom.convertPathToMarquee()`

Availability
Fireworks 7.

Usage
```javascript
dom.convertPathToMarquee(mode, featherAmount)
```

Arguments
- `mode` Sets the mode. Acceptable values are "hard edge", "antialias", and "feather".
- `featherAmount` Sets the amount of feathering for the marquee selection. This value is ignored if `mode` is not set to "feather".

Returns
Nothing.

Description
Converts path to marquee selection.

`dom.convertToAnimSymbol()`

Availability
Fireworks 4.

Usage
```javascript
dom.convertToAnimSymbol(name, numFrames, offsetDistPt, rotationAmount, scaleAmount, startOpacity, endOpacity)
```

Arguments
- `name` A string that specifies a name for the new animation symbol.
- `numFrames` An integer that specifies the number of frames through which the symbol animates.
- `offsetDistPt` A point that specifies the distance the animation will move in pixels (see “Point data type” on page 9). For example, passing `{x:100, y:25}` animates the symbol to the right 100 pixels and 25 pixels down.
- `rotationAmount` A floating-point value that specifies the degrees of rotation to be applied to the animation symbol. For example, passing a value of 720 specifies an animation that does two complete clockwise rotations. To rotate the animation counter-clockwise, pass a negative number.
- `scaleAmount` A positive floating-point value that specifies the amount of scaling to be applied to the animation symbol. For example, passing a value of 50 scales the symbol to 50% of its current size, and passing 200 scales it to twice its current size. To specify no scaling, pass 100.
- `startOpacity` and `endOpacity` Float values between 0 and 100 that specify the starting and ending opacity for the animation symbol.
Returns
Nothing.

Description
Converts the selected items to a new animation symbol.

See also
dom.convertAnimSymbolToGraphicSymbol(), dom.convertToSymbol(), dom.setAnimInstanceNumFrames()

dom.convertToPaths()

Availability
Fireworks 3.

Usage
dom.convertToPaths()

Arguments
None.

Returns
Nothing.

Description
Converts the selected text items into editable paths.

Example
The following command converts the selected text items into editable paths:

fw.getDocumentDOM().convertToPaths();

dom.convertToSymbol()

Availability
Fireworks 3.

Usage
dom.convertToSymbol(type, name, status)

Arguments
type  Acceptable values are "graphic", "button", and "animation".
name  A name for the new symbol.
status A Boolean value that toggles 9-slice scaling between enabled and disabled.

Returns
Nothing.
Description
Converts the selected items to a new symbol.

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

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

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

**dom.convolveSelection()**

Availability
Fireworks MX 2004.

Usage
```
dom.convolveSelection(kernelWidth, kernelHeight, kernelValues, affectsAlpha)
```

Arguments
- `kernelWidth`: An integer that defines the width of the filter coefficients.
- `kernelHeight`: An integer that defines the height of the filter coefficients.
- `kernelValues`: An array of integers that defines the values for specific filter patterns.
- `affectsAlpha`: A Boolean value: `true` means the convolution filter affects the transparency of the bitmap; `false` means that the bitmap transparency isn’t affected by the filter.

Returns
Nothing

Description
Applies convolution, or irregular, filters to the selected bitmap based on the pattern defined by the argument values.

Example
The following example applies an edge-detection filter to the bitmap:

```
// width of convolution kernel
var w = 3;
// height of convolution kernel
var h = 3;
// Edge detection kernel
var k = new Array(0, 1, 0, 1, -4, 1, 0, 1, 0);
fw.getDocumentDOM().convolveSelection(w, h, k, false);
```
**dom.copyHtmlWizard()**

**Availability**
Fireworks MX.

**Usage**

```javascript
dom.copyHtmlWizard()
```

**Arguments**
None.

**Returns**
Nothing.

**Description**
Opens the Copy HTML Wizard dialog box.

**Example**
The following command opens the Copy HTML Wizard dialog box:

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

**dom.copyToHotspot()**

**Availability**
Fireworks 3.

**Usage**

```javascript
dom.copyToHotspot(hotspotType, {whatIfMultipleSelected}, {makeRectangular})
```

**Arguments**

- **hotspotType** Acceptable values are "hotspot" and "slice".

- **whatIfMultipleSelected** 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.

- **makeRectangular** An optional Boolean value that determines if the slice for the hotspot will be a rectangle or polygon. If true (the default), Fireworks creates a rectangular slice; otherwise, the slice is a polygon if the shape being copied to the slice is a polygon.

**Returns**
Nothing.

**Description**
Creates one or more hotspots from the selection.
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.

**Usage**
dom.cropSelection(boundingRectangle)

**Arguments**
boundingRectangle A rectangle that specifies the bounds within which the selection should be cropped (see “Rectangle data type” on page 9).

**Returns**
Nothing.

**Description**
Crops the selection to the specified rectangle.

**dom.deleteAllInDocument()**

**Availability**
Fireworks MX.

**Usage**
dom.deleteAllInDocument()

**Arguments**
None.

**Returns**
Nothing.

**Description**
Deletes all of the objects in the document.

**dom.deleteFrames()**

**Availability**
Fireworks 3.
Usage
dom.deleteFrames(frameIndex, howMany)

Arguments
frameIndex An integer value that specifies the location at which to begin deleting frames, starting with 0 (although, to specify the current frame, pass –1).
howMany Specifies how many frames to delete.

Returns
Nothing.

Description
Deletes one or more frames.

dom.deleteLayer()

Availability
Fireworks 3.

Usage
dom.deleteLayer(layerIndex)

Arguments
layerIndex An integer value that specifies the layer to be deleted, starting with 0 (although, to specify the current layer, pass –1 here).

Returns
Nothing.

Description
Deletes a layer.

Example
The following command deletes the current layer:
fw.getDocumentDOM().deleteLayer(-1);

dom.deletePageAt()

Availability
Fireworks CS3.

Usage
dom.deletePageAt(pageNum)
Arguments
pageNum A long value that indicates the page number of the page to be deleted.

Returns
Nothing.

Description
Deletes a specified page from the current document. For example:
fw.getDocumentDOM().deletePageAt(0)

dom.deletePointOnPath()

Availability
Fireworks 4.

Usage
dom.deletePointOnPath(contourIndex, pointIndex)

Arguments
contourIndex An integer value that specifies the contour that contains the point to be deleted, starting with 0 (although, to specify the current contour, pass –1 here).
pointIndex An integer value that specifies the point to be deleted, starting with 0 (although, to specify the current point, pass –1 here).

Returns
Nothing.

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.

Example
The following command deletes the currently selected point:
fw.getDocumentDOM().deletePointOnPath(-1, -1);

dom.deleteSavedSelection()

Availability
Fireworks 3.

Usage
dom.deleteSavedSelection(selection)
Arguments
selection The name of the saved bitmap selection.

Returns
Nothing.

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

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().deleteSavedSelection(false);

\texttt{dom.deleteSelection()}

Availability
Fireworks 3.

Usage
\texttt{dom.deleteSelection(bFillDeletedArea)}

Arguments
\texttt{bFillDeletedArea} This argument is ignored if Fireworks is not in bitmap mode. If Fireworks is in bitmap mode and \texttt{bFillDeletedArea} is \texttt{true}, the deleted pixels are filled with the current fill color. If \texttt{false}, the deleted pixels are filled to transparent.

Returns
Nothing.

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

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);

\texttt{dom.deleteSymbol()}

Availability
Fireworks 3.

Usage
\texttt{dom.deleteSymbol(symbolName)}
Arguments
symbolName  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.

Description
Deletes the specified symbols from the library.

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.

Usage
dom.detachInstanceFromSymbol()

Arguments
None.

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

Returns
Nothing.

dom.detachTextFromPath()

Availability
Fireworks 3.

Usage
dom.detachTextFromPath()

Arguments
None.
Returns
Nothing.

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

**dom.detachSharedLayer()**

Availability
Fireworks CS3.

Usage
```javascript
dom.detachSharedLayer(layerNum, pageNum)
```

Arguments
- `layerNum` A long value that specifies the layer number for the layer that is to be detached.
- `pageNum` A long value that specifies the page number of the page from which the layer will be detached.

Returns
Nothing.

Description
Detaches the specified shared layer from the specified page. You can only detach a parent layer, not a sub layer. When the parent later is detached, the sub layers are automatically detached as well.

Example:
```javascript
fw.getDocumentDOM().detachSharedLayer(1, 1)
```

**dom.distribute()**

Availability
Fireworks 3, updated with `distributeToCanvas` parameter in Fireworks 8.

Usage
```javascript
dom.distribute(distmode, distributeToCanvas)
```

Arguments
- `distmode` Acceptable values are "vertical" and "horizontal".
- `distributeToCanvas` A Boolean value that determines whether items are distributed to the canvas. Default value is "false".

Returns
Nothing.
Description
Distributes the selection along a vertical or horizontal dimension.

dom.distributeLayerToFrames()

Availability
Fireworks 3.

Usage
dom.distributeLayerToFrames(layerIndex)

Arguments
layerIndex An integer value that specifies the layer that contains the items to be distributed, starting with 0 (although, to specify the current layer, pass –1 here).

Returns
Nothing.

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.

dom.distributeSelectionToFrames()

Availability
Fireworks 3.

Usage
dom.distributeSelectionToFrames()

Arguments
None.

Returns
Nothing.

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.

dom.dragControlPoint()

Availability
Fireworks MX 2004.
Usage
	
dom.dragControlPoint(index, newLoc, shiftKeyDown, ctrlCmdKeyDown, altOptKeyDown)

Arguments
	index  The index of the control point to move.
	newLoc  Specifies the new location of the point.
	shiftKeyDown  Specifies whether the Shift key is pressed.
	ctrlCmdKeyDown  Specifies whether the Control key (Windows) or Command key (Macintosh) is pressed.
	altOptKeyDown  Specifies whether the Alt key (Windows) or Option key (Macintosh) is pressed.

Returns
	Nothing.

Description

Drags the specified control point to the new location.

dom.duplicateFrame()  

Availability
	Fireworks 3.

usage

dom.duplicateFrame(frameIndex, howMany, where, bDupeSelectionOnly)

Arguments
	frameIndex  An integer value that specifies the frame to duplicate, starting with 0 (although, to specify the current frame, pass −1 here).
	howMany  An integer that specifies how many copies of the frame to make.
	where  Acceptable values are "beginning", "before current", "after current", and "end".
	bDupeSelectionOnly  If bDupeSelectionOnly is true, only items in the specified frame that are selected are duplicated to the new frame.

Returns
	Nothing.

Description

Duplicates a frame.

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.

**Usage**
```
dom.duplicateLayer(layerIndex, {howMany}, {where})
```

**Arguments**
- `layerIndex`: An integer value that specifies the layer to duplicate, starting with 0 (although, to specify the current layer, pass -1 here).
- `howMany`: An optional integer that specifies how many times to duplicate the layer. If omitted, the layer is duplicated once.
- `where`: 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.

**Description**
Duplicates a layer.

**Example**
The following command places three copies of the current layer at the end of the document:
```
fw.getDocumentDOM().duplicateLayer(-1, 3, "end");
```

**dom.duplicatePage()**

**Availability**
Fireworks CS3.

**Usage**
```
dom.duplicatePage(pageNum)
```

**Arguments**
- `pageNum`: An long value that specifies the page number of the page to be duplicated.

**Returns**
Nothing.

**Description**
Duplicates a page. For example:
```
fw.getDocumentDOM().duplicatePage(1)
```
extending fireworks

the document object

dom.duplicateSelection()

availability
fireworks 3.

usage
dom.duplicateSelection()

arguments
none.

returns
nothing.

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

example
the following command duplicates the selected items:

fw.getDocumentDOM().duplicateSelection();

see also
dom.cloneSelection()

dom.duplicateSelectionToFrameRange()

availability
fireworks 3.

usage
dom.duplicateSelectionToFrameRange(frameIndexFirst, frameIndexLast)

arguments
frameIndexFirst and frameIndexLast  integer values that specify the range of frames (inclusive) to which the items should be copied, starting with 0 (although, to specify the current frame, pass -1 here).

•  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.

description
Duplicates the selection to a range of frames of the document.
**dom.duplicateSelectionToFrames()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.duplicateSelectionToFrames(whichFrames)
```

**Arguments**
- `whichFrames` Acceptable values 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.

**Description**
Duplicates the selection to specified frames of the document.

**dom.duplicateSymbol()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.duplicateSymbol(symbol)
```

**Arguments**
- `symbol` 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.

**Description**
Duplicates the specified symbol.

**dom.duplicateSymbolForAlias()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.duplicateSymbolForAlias()
```
Arguments
None.

Returns
Nothing.

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.)

dom.elementsAt()

Availability
Fireworks MX 2004.

Usage
dom.elementsAt(where)

Arguments
where Specifies which rectangle to check for elements. To find the elements under a single point (similar to selecting with the Subselection tool), set left equal to right and top equal to bottom. To find elements within a rectangle (similar to drag-selecting with the Pointer tool), set the values to the desired rectangle.

Returns
An array of zero or more elements.

Description
Returns a list of zero or more elements at the given location. Similar to selecting with the Subselection tool or drag-selecting with the Pointer tool.

dom.enableElementMask()

Availability
Fireworks 4, updated with new arguments in Fireworks MX.

Usage
dom.enableElementMask(enable, selectAndEnterPaintModeIfPossible, {newSelectionMask})

Arguments
enable A Boolean value that toggles the element mask between enabled (true) and disabled (false).

selectAndEnterPaintModeIfPossible 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  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. This argument is null by default.

Returns  Nothing.

Description  Enables or disables the element mask on the selected element. If more than one element (or no elements) are selected when this function is called, Fireworks throws an exception.

dom.enableNineScale()  
Availability  Fireworks CS3

Usage  
dom.enableNineScale(status)

Arguments  
status  A Boolean value that toggles 9-slice scaling between enabled and disabled.

Returns  Nothing.

Description  Enables or disables 9-slice scaling for the selected symbol.

Example  The following command enables 9-slice scaling for the selected symbol:

fw.getDocumentDOM().enableNineScale(true);

dom.enableTextAntiAliasing()  
Availability  Fireworks MX.

Usage  
dom.enableTextAntiAliasing(antiAlias)

Arguments  
antiAlias  A Boolean value to turn anti-aliasing on (true) or off (false).

Returns  Nothing.
Description
Turns anti-aliasing on or off for the selected blocks of text.

`dom.enterPaintMode()`

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

Usage
```
dom.enterPaintMode({newSelectionMask})
```

Arguments
`newSelectionMask` An optional bitmap selection mask. When `newSelectionMask` is not `null`, the selection is set on the currently selected bitmap after entering paint mode. This argument is `null` by default.

Returns
Nothing.

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

`dom.exitPaintMode()`

Availability
Fireworks 3.

Usage
```
dom.exitPaintMode()
```

Arguments
None.

Returns
Nothing.

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

`dom.exitSymbolEdit()`

Availability
Fireworks 9.

Usage
```
dom.exitSymbolEdit(level)
```

Description
Leaves bitmap mode. Has no effect if Fireworks is not in bitmap mode.
Arguments
level Integer that specifies the number of levels back to exit symbol edit mode, especially when exiting Nested Symbols.

Returns
Nothing.

Description
Exits the symbol edit mode through the number of levels specified.

dom.exportElements()

Availability
Fireworks 10.

Usage
dom.exportElements(elements, imagesUrl, name)

Arguments
elements An array that contains the objects to be exported.
imagesUrl Folder name to which the image is exported. Specified as file:///.
name Name of the image.

Returns
Nothing.

Description
Exports an array of elements on the canvas to a 32-bit PNG image, based on the image export settings.

dom.exportOptions.loadColorPalette()

Availability
Fireworks 3.

Usage
dom.exportOptions.loadColorPalette(fileURL)

Arguments
fileURL 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.
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 object" on page 260.

**dom.exportOptions.saveColorPalette()**

**Availability**
Fireworks 3.

**Usage**
dom.exportOptions.saveColorPalette(fileURL)

**Arguments**
fileURL 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.

**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 object” on page 260.

**dom.exportTo()**

**Availability**
Fireworks 3.

**Usage**
dom.exportTo(fileURL, {exportOptions})

**Arguments**
fileURL A string, which is expressed as a file://URL, that specifies the name of the exported file.
exportOptions An ExportOptions object (see “ExportOptions object” on page 260). This argument is optional. 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.

**Description**
Exports the document as specified.
dom.fillSelectedPixels()

Availability
Fireworks 3.

Usage
dom.fillSelectedPixels(clickPt, p1, p2, p3, bFillSelectionOnly, tolerance, edgemode, featherAmt)

Arguments
- **clickPt** A point that specifies the x,y coordinates of the pixel to be filled or generated (see “Point data type” on page 9).
- **p1, p2, and p3** Points that specify the fill-vector. These arguments are ignored if the current fill does not use a fill-vector.
- **bFillSelectionOnly** 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** An integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- **edgemode** Acceptable values are “hard edge”, “antialias”, and “feather”.
- **featherAmt** An integer between 0 and 32,000, inclusive, that specifies the number of pixels to feather. This value is ignored if the value of edgemode is not “feather”.

Returns
Nothing.

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.

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

```javascript
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.

Usage
dom.filterSelection(LiveEffect)

Arguments
- **LiveEffect** An Effect object (see “Effect object” on page 248).
Returns
Nothing.

Description
Applies the specified pixel filter to the selection. Items that are not images 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()`.

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

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

dom.filterSelectionByID()

Availability
Fireworks 8.

Usage
dom.filterSelectionByID(ID)

Arguments

- **ID** The EffectMoaID of the filter you want applied.

Returns
Nothing.

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.

dom.filterSelectionByName()

Availability
Fireworks 3.

Usage
dom.filterSelectionByName(category, name)

Arguments

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

Returns
Nothing.

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.

`dom.findNamedElements()`

Availability
Fireworks 4.

Usage
`dom.findNamedElements(name)`

Arguments
name  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.

Description
Looks for elements that have the specified name.

See also
`dom.setElementName()`

`dom.flattenDocument()`

Availability
Fireworks 3.

Usage
`dom.flattenDocument()`

Arguments
None.

Returns
Nothing.

Description
Flattens the entire document into a single pixel image. This is the same behavior as the Merge Layers command.
**dom.flattenSelection()**

**Availability**
Fireworks 3.

**Usage**
`dom.flattenSelection()`

**Arguments**
None.

**Returns**
Nothing.

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

**dom.getFontMarkup()**

**Availability**
Fireworks 3.

**Usage**
`dom.getFontMarkup(fontAttribute)`

**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.

**Description**
 Gets a font markup attribute for the selected text.

**dom.getLockGuides()**

**Availability**
Fireworks 8.

**Usage**
`dom.getLockGuides()`

**Arguments**
None.
Returns
True if the guides are locked; false otherwise.

Description
Determines if the guides are locked.

**dom.getPageSetter()**

Availability
Fireworks 9.

Usage
dom.getPageSetter()

Arguments
None.

Returns
The PageSetter object which helps in changing pages without refreshing.

Description
Gets the pageSetter object of type SetCurrentPage, with the pageNum, which is a zero-based index of the current page.

**dom.getParentLayerNum()**

Availability
Fireworks CS3.

Usage
dom.getParentLayerNum(currentLayer)

Arguments
currentLayer A long value that specifies the index of the current layer.

Returns
The layer index number of the parent layer. If the specified layer is a top-level layer, it returns a value of -1.

Description
Gets the parent layer index number for the specified layer.

**dom.getPixelMask()**

Availability
Fireworks 3, deprecated in 4.
Usage

`dom.getPixelMask()`

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 data type” on page 8.

Description
Gets the current pixel-selection mask. The result of this call could be used to call `dom.enableElementMask()` or `dom.enterPaintMode()`.

`dom.getSelectionBounds()`

Availability
Fireworks 3.

Usage

`dom.getSelectionBounds()`

Arguments
None.

Returns
A rectangle (see “Rectangle data type” on page 9). Returns `null` if nothing is selected.

Description
Gets the bounding rectangle of the selection.

`dom.getShowGrid()`

Availability
Fireworks 3.

Usage

`dom.getShowGrid()`

Arguments
None.

Returns
`true` if the grid is visible; `false` otherwise.
Description
Determines whether the grid is visible.

dom.getShowGuides()

Availability
Fireworks 8.

Usage
dom.getShowGuides()

Arguments
None.

Returns
true if the guides are visible; false otherwise.

Description
Determines if the guides are visible.

dom.getShowRulers()

Availability
Fireworks 3.

Usage
dom.getShowRulers()

Arguments
None.

Returns
true if the rulers are visible; false otherwise.

Description
Determines whether the rulers are visible.

dom.getSnapToGuides()

Availability
Fireworks 8.

Usage
dom.getSnapToGuides()
Arguments
None.

Returns
ture if the Snap to Guides function is available; false otherwise.

Description
Determines if the Snap to Guides function is available.

**dom.getSnapToGrid()**

Availability
Fireworks 3.

Usage
dom.getSnapToGrid()

Arguments
None.

Returns
ture if the Snap to Grid function is active; false otherwise.

Description
Determines whether the Snap to Grid function is active.

**dom.getTextAlignment()**

Availability
Fireworks 3.

Usage
dom.getTextAlignment()

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.

Description
Gets the alignment of selected text.
**dom.group()**

**Availability**
Fireworks 3, argument deprecated in 4.

**Usage**
```
dom.group({type})
```

**Arguments**
- **type** 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. In Fireworks 4, "mask to image" and "mask to path" are deprecated.

**Returns**
Nothing.

**Description**
Groups the selection. To ungroup elements use `dom.ungroup()` (see `dom.ungroup()`).

**Example**
The following command sets the selected group to mask to the image:
```
replace with fw.getDocumentDOM().group("normal");
```

**dom.hasCharacterMarkup()**

**Availability**
Fireworks 3, updated in Fireworks 4.

**Usage**
```
dom.hasCharacterMarkup(tag)
```

**Arguments**
- **tag** Acceptable values 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.

**Description**
Determines whether the selected text has the specified character markup.

**dom.hasMasterPage()**

**Availability**
Fireworks CS3.
Usage

```javascript
dom.hasMasterPage()
```

Arguments

None.

Returns

A Boolean value of `true` if the current document has a master page, or `false` if there is no master page.

Description

Indicates whether or not a master page exists for the specified document. For example:

```javascript
fw.getDocumentDOM().hasMasterPage()
```

---

**dom.hideSelection()**

Availability

Fireworks 3.

Usage

```javascript
dom.hideSelection()
```

Arguments

None.

Returns

Nothing.

Description

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

---

**dom.importFile()**

Availability

Fireworks 3.

Usage

```javascript
dom.importFile(fileURL, boundingRectangle, bMaintainAspectRatio)
```

Arguments

- `fileURL` The filename of the file to be imported, which is expressed as a `file://URL`.
- `boundingRectangle` A rectangle that specifies the size to make the imported file (see "Rectangle data type" on page 9). 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 If 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.

Description
Imports the specified file at the specified location.

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

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

dom.importSymbol()

Availability
Fireworks 3.

Usage
dom.importSymbol(fileURL, bAddToDoc, bAllowUI)

Arguments
- `fileURL` The name of the file to be imported into the library, which is expressed as a `file://URL`.
- `bAddToDoc` 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.
- `bAllowUI` 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.

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

dom.importSymbolButNotAsAlias()

Availability
Fireworks MX.

Usage
dom.importSymbolButNotAsAlias(filepath, whichSymbol)
Arguments
filepath  The file URL of the file that contains the symbol to be copied.
whichSymbol  The index of the symbol within the document, which is specified in the filepath.

Returns
Nothing.

Description
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.

dom.inLaunchAndEdit()

Availability
Fireworks MX.

Usage
dom.inLaunchAndEdit()

Arguments
None.

Returns
A Boolean value: true if opened by a launch-and-edit operation; false otherwise.

Description
Specifies whether document was opened by a launch-and-edit operation.

dom.insertPointInPath()

Availability
Fireworks 3.

Usage
dom.insertPointInPath(contourIndex, ptToInsertBefore, tParameter, controlPointFirst, mainPoint, controlPointLast)

Arguments
contourIndex  A zero-based index that specifies the contour into which the Bézier point is inserted. For paths with multiple contours, the contours are in an arbitrary order.
ptToInsertBefore A zero-based index 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 A floating-point value between 0 and 1 that specifies where to insert the new point in the Bézier segment.

controlPointFirst, mainPoint, and controlPointLast 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 data type” on page 9).

Returns
Nothing.

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.

See also
dom.appendPointToPath()

dom.insertSmartShapeAt()

Availability
Fireworks MX 2004.

Usage
dom.insertSmartShapeAt(name, location, useToolBlendModeOpacity)

Arguments
name A string specifying the name of the Auto Shape.
location The upper-left point of the Auto Shape.
useToolBlendModeOpacity Determines whether the new shape object should have the blend mode and opacity settings set for the Auto Shape Tools (set by the user in the Property inspector), or use standard values. The bUseToolBlendModeOpacity argument is a Boolean value: true if the shape will use the blend mode and opacity set for the Auto Shape Tools; false if the shape will use the standard values (alpha blend mode and 100% opacity).

Returns
Nothing.

Description
Inserts an Auto Shape at the specified location.

dom.insertSymbolAt()

Availability
Fireworks 8.
Usage
`dom.insertSymbolAt(uiName, locationPoint)`

Arguments
`uiName` The name of the symbol in the library. If more than one symbol exists with the specified name, Fireworks inserts the first symbol named.

`locationPoint` The center of the symbol expressed as x, y coordinates.

Returns
Nothing.

Description
Inserts a symbol instance at the specified location.

`dom.insertText()`

Availability
Fireworks 8.

Usage
`dom.insertText(text)`

Arguments
`text` The text to insert.

Returns
Nothing.

Description
Inserts the given text into a selected text block at the current text insertion point. If no text is selected it does nothing.

`dom.isMasterPageLayer()`

Availability
Fireworks CS3.

Usage
`dom.isMasterPageLayer(layerNum)`

Arguments
`layerNum` A long value that specifies the layer number.

Returns
A Boolean value: `true` if the specified layer is a master page layer; `false` otherwise.
The Document object

Description
Indicates whether or not the specified layer is a master page layer. For example:

```
fw.getDocumentDOM().isMasterPageLayer(0)
```

**dom.isSelectionDirectlyAboveBitmapObject()**

**Availability**
Fireworks MX.

**Usage**
```
dom.isSelectionDirectlyAboveBitmapObject()
```

**Arguments**
None.

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

**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.

**dom.joinPaths()**

**Availability**
Fireworks 3.

**Usage**
```
dom.joinPaths()
```

**Arguments**
None.

**Returns**
Nothing.

**Description**
Joins the selected paths.

**dom.knifeElementsFromPoint()**

**Availability**
Fireworks 3.
Usage

`dom.knifeElementsFromPoint(from, tolerance)`

Arguments

- `from` A point that specifies the x,y coordinates of the point that the user clicked (see “Point data type” on page 9).
- `tolerance` A floating-point value ≥ 0 that specifies the tolerance within which items are cut.

Returns

A Boolean value: `true` if anything was cut; `false` otherwise.

Description

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

See also

`dom.knifeElementsFromPoints()`

`dom.knifeElementsFromPoints()`

Availability

Fireworks 3.

Usage

`dom.knifeElementsFromPoints(from, to, tolerance)`

Arguments

- `from` A point that specifies the x,y coordinates of the point where the user clicked and started to drag (see “Point data type” on page 9).
- `to` A point that specifies the x,y coordinates of the point where the user ended the drag operation.
- `tolerance` A floating-point value ≥ 0 that specifies the tolerance within which items are cut.

Returns

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

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.

See also

`dom.knifeElementsFromPoint()`

`dom.linkElementMask()`

Availability

Fireworks 4.
EXTENDING FIREWORKS
The Document object

Usage

```javascript
DOM.linkElementMask(frame, layer, element, bLink)
```

Arguments

- `frame`: An integer value that specifies the frame that contains the element, starting with 0 (although, to specify the current frame, pass -1 here).
- `layer`: An integer value that specifies the layer that contains the element, starting with 0 (although, to specify the current layer, pass -1 here).
- `element`: An integer value that specifies the element, starting with 0 (although, to specify the current element, pass -1 here).
- `bLink`: If `true`, the element masks are linked to their elements; if `false`, they are unlinked from their elements.

Returns

Nothing.

Description

Links or unlinks the element mask on the selected element. If more than one element (or no elements) are selected when this function is called, Fireworks throws an exception. An exception is also thrown if the element has no element mask.

```javascript
DOM.lockNineScale(status)
```

Availability

Fireworks CS3

Usage

```javascript
DOM.lockNineScale(status)
```

Arguments

- `status`: A Boolean value that toggles 9-slice scaling between locked and unlocked.

Returns

Nothing.

Description

Locks or unlocks 9-slice scaling guides for the selected symbol.

Example

The following command locks 9-slice scaling guides for the selected symbol:

```javascript
fw.getDocumentDOM().lockNineScale(true);
```

```javascript
DOM.lockSelection()
```

Availability

Fireworks 8.
Usage

dom.lockSelection()

Arguments
None.

Returns
Nothing.

Description
Locks the selection.

dom.makeFind()

Availability
Fireworks 3.

Usage

dom.MakeFind(findSpec)

Arguments
findSpec A Find object (see “Find object” on page 20).

Returns
A Find object.

Description
Creates an object of class Find to perform a search-and-replace operation in a document.

dom.makeActive()

Availability
Fireworks 3.

Usage

dom.makeActive()

Arguments
None.

Returns
Nothing.

Description
Makes the selected document active for editing.
dom.mergeDown()

Availability
Fireworks MX.

Usage
dom.MergeDown()

Arguments
None.

Returns
Nothing.

Description
Merges selected objects to the bitmap directly below the selected objects. Succeeds only if the object immediately below the selection is a bitmap. For more information, see dom.isSelectionDirectlyAboveBitmapObject().

dom.modifyPointOnPath()

Availability
Fireworks 3.

Usage
dom.modifyPointOnPath(contourIndex, ptToModify, controlPointFirst, mainPoint, controlPointLast, dReapplyAttrs, bClosePath)

Arguments
contourIndex A zero-based index that specifies the contour into which the Bézier point is inserted. For paths with multiple contours, the contours are in an arbitrary order.

ptToModify A zero-based index that specifies the point to be modified.

controlPointFirst, mainPoint, and controlPointLast 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 data type" on page 9).

dReapplyAttrs 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.

bClosePath 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.

Description
Modifies an existing point on the selected path.
**dom.motionBlurSelection()**

**Availability**
Fireworks MX 2004.

**Usage**

```
dom.motionBlurSelection(typeStr, angle, distance, samples)
```

**Arguments**
- **typeStr** A string that specifies the type of blur to apply. Valid values are "linear", "radial", and "zoom".
- **angle** An integer between 0 and 359 that specifies in degrees the direction of the blur, similar to the drop shadow effect angle.
- **distance** A floating-point value between 0 and 400 that specifies in pixels how far from the original image the blur effect will extend.
- **samples** An integer that defines the number of times the original image is cloned and blurred to produce the desired effect.

**Returns**
Nothing.

**Description**
Applies the Motion Blur effect (same as selecting the Filters > Blur > Motion Blur menu option) to the selection.

**dom.moveBezierHandleBy()**

**Availability**
Fireworks 3.

**Usage**

```
dom.moveBezierHandleBy(whichPath, contourIndex, ptToModify, deltaControlPointFirst, deltaControlPointLast)
```

**Arguments**
- **whichPath** A zero-based index that specifies an index into the list of selected items, indicating which item contains the Bézier handles to move.
- **contourIndex** A zero-based index that specifies the contour that contains the handles to move. For paths with multiple contours, the contours are in an arbitrary order.
- **ptToModify** A zero-based index that specifies the point whose handles are moved.
- **deltaControlPointFirst** and **deltaControlPointLast** 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 1 pixel and down by 2 pixels.

**Returns**
Nothing.
**dom.moveElementMaskBy()**

**Availability**
Fireworks 4.

**Usage**

```javascript
dom.moveElementMaskBy(delta)
```

**Arguments**

- `delta` A point that specifies the x,y coordinate values by which the element masks are moved (see “Point data type” on page 9). For example, passing `({x:1,y:2})` moves the element masks 1 pixel to the right and 2 pixels down.

**Returns**
Nothing.

**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.

**dom.moveFillVectorHandleBy()**

**Availability**
Fireworks 3.

**Usage**

```javascript
dom.moveFillVectorHandleBy(delta, whichHandle, bConstrain, bMoveJustOne)
```

**Arguments**

- `delta` A point that specifies the x,y coordinate values by which the handle is moved (see “Point data type” on page 9). For example, passing `({x:1,y:2})` specifies a location that is right by 1 pixel and down by 2 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.

- `bConstrain` Boolean. If true, movement is constrained to 45° increments.

- `bMoveJustOne` Boolean. If true, only the specified handle moves. If false, other handles might move in sync when the specified handle is moved.

**Returns**
Nothing.
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.

**dom.moveMaskGroupContentsBy()**

**Availability**
Fireworks 3.

**Usage**
`dom.moveMaskGroupContentsBy(delta)`

**Arguments**
- **delta**: A point that specifies the x,y coordinate values by which the element is moved (see "Point data type" on page 9). For example, passing (1,2) moves the element 1 pixel to the right and 2 pixels down.

**Returns**
Nothing.

**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.

See also
- `dom.moveElementMaskBy()`

**dom.moveNineScaleGuide()**

**Availability**
Fireworks 10.

**Usage**
`dom.moveNineScaleGuide(delta, guidekind)`

**Arguments**
- **delta**: A point that specifies the values by which the 9-slice scaling guide is moved.
- **guidekind**: Direction in which the 9-slice scaling guide is moved.

**Returns**
Nothing.

**Description**
Moves a 9-slice scaling guide’s position by specified pixels. Acceptable values for guidekind are "horizontal-top", "horizontal-bottom", "vertical-left" and "vertical-right". If guidekind is "horizontal", it is assumed that the specified positions are y coordinates; if it is "vertical", it is assumed that the specified positions are x coordinates.
Example
The following command moves a horizontal top guide by 10 pixels.

```
fw.getDocumentDOM().moveNineScaleGuide("horizontal-top",10);
```

dom.movePixelMaskBy()

Availability
Fireworks 4.

Usage
dom.movePixelMaskBy(delta)

Arguments
delta A point that specifies the x,y coordinate values by which the bitmap mode selection is moved (see “Point data type” on page 9). For example, passing ({x:1,y:2}) moves the bitmap mode selection 1 pixel to the right and 2 pixels down.

Returns
Nothing.

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

dom.movePointOnHotspotBy()

Availability
Fireworks 3.

Usage
dom.movePointOnHotspotBy(ptToModifyIndex, delta)

Arguments
ptToModifyIndex A zero-based index that specifies which point on the path is to move.
delta A point that specifies the x,y coordinate values by which the point is moved (see “Point data type” on page 9). For example, passing ({x:1,y:2}) moves the point 1 pixel to the right and 2 pixels down.

Returns
Nothing.

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.
dom.movePointOnHotspotByWithFlags()

Availability
Fireworks MX.

Usage
dom.MovePointOnHotspotByWithFlags(ptToModifyIndex, delta, flags)

Arguments
ptToModifyIndex A zero-based index that specifies which point on the path is to move.
delta A point that specifies the x-,y-coordinate values by which the point is moved (see “Point data type” on page 9). For example, passing ({x:1,y:2}) moves the point 1 pixel to the right and 2 pixels down.
flags 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.

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.

dom.moveSelectedBezierPointsBy()

Availability
Fireworks 3.

Usage
dom.moveSelectedBezierPointsBy(delta)

Arguments
delta A point that specifies the x,y coordinate values by which the selected Bézier points are moved (see “Point data type” on page 9). For example, passing ({x:1,y:2}) moves the Bézier points 1 pixel to the right and 2 pixels down.

Returns
Nothing.

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.

dom.moveSelectionBy()

Availability
Fireworks 3.
Usage

`dom.moveSelectionBy(delta, bMakeCopy, doSubSel)`

Arguments

`delta` A point that specifies the x,y coordinate values by which the selection moved (see "Point data type" on page 9). For example, passing `{x:1, y:2}` moves the selection 1 pixel to the right and 2 pixels down.

`bMakeCopy` The items that are copied instead of moved.

`doSubSel` 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.

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.

Example

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

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

dom.moveSelectionMaskBy()

Availability

Fireworks 4.

Usage

`dom.moveSelectionMaskBy(delta)`

Arguments

`delta` A point that specifies the x,y-coordinate values by which the mask is moved (see "Point data type" on page 9). For example, passing `{x:1,y:2}` moves the mask 1 pixel to the right and 2 pixels down.

Returns

Nothing.

Description

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

dom.moveSelectionTo()

Availability

Fireworks 3.
Usage
```
dom.moveSelectionTo(location, bMakeCopy, doSubSel)
```

Arguments
- **location**: A point that specifies the x-,y-coordinate values of the location to which the selection is moved or copied (see “Point data type” on page 9).
- **bMakeCopy**: Specifies copying instead of moving the selection.
- **doSubSel**: 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.

Description
Moves or copies the selection to the specified location.

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.

Usage
```
dom.moveSelectionToFrame(frameIndex, bMakeCopy)
```

Arguments
- **frameIndex**: An integer value that specifies the frame to which the selection is moved or copied, starting with 0 (although, to specify the current frame, pass –1 here).
- **bMakeCopy**: If bMakeCopy is true, the selection is copied instead of moved.

Returns
Nothing.

Description
Moves or copies the selection to the specified frame.

**dom.moveSelectionToLayer()**

Availability
Fireworks 3, updated in Fireworks 4.
Usage

dom.moveSelectionToLayer(layerIndex, bMakeCopy, {whatIfMultipleSelected}, {elementIndex})

Arguments

layerIndex An integer value that specifies the layer to which the selection should be moved or copied, starting with 0 (although, to specify the current layer, pass –1 here).

bMakeCopy Boolean. If true, the selection is copied instead of moved.

whatIfMultipleSelected 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 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 A zero-based index, added in Fireworks 4, 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.

Description

Moves or copies the selection to the specified layer.

**dom.moveSelectionToNewLayer()**

Availability

Fireworks 3.

Usage

dom.moveSelectionToNewLayer(bMakeCopy)

Arguments

bMakeCopy Boolean. If true, the selected items are copied instead of moved.

Returns

Nothing.

Description

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

**dom.pagesCount**

Availability

Fireworks CS3.
Usage
```
dom.pagesCount
```

Arguments
None.

Returns
An integer returning total number of pages in the current document.

Description
Indicates the number of pages in the current document. For example:
```
fw.getDocumentDOM().pagesCount
```

---

**dom.pageName()**

Availability
Fireworks CS3.

Usage
```
dom.pageName()
```

Arguments
None.

Returns
A string specifying the name of the current page.

Description
Indicates the name of the current page. For example:
```
fw.getDocumentDOM().pageName
```

---

**dom.pathCrop()**

Availability
Fireworks 3.

Usage
```
dom.pathCrop()
```

Arguments
None.

Returns
Nothing.
Description
Performs a crop operation on the selected paths.

**dom.pathExpand()**

**Availability**
Fireworks 3.

**Usage**
```
dom.pathExpand(width, miter, cap, join)
```

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

**Returns**
Nothing.

**Description**
Performs an expand operation on the selected paths.

**dom.pathInset()**

**Availability**
Fireworks 3.

**Usage**
```
dom.pathInset(width, miter, join)
```

**Arguments**
- **width** A floating-point value that specifies the new width of the selected paths, in pixels.
- **miter** A floating-point value that specifies the new miter angle of the selected paths, in pixels. This argument is ignored if the value of **join** is not "miter".
- **join** Acceptable values are "bevel", "round", and "miter".

**Returns**
Nothing.

**Description**
Performs an inset operation on the selected paths.
dom.pathIntersect()

Availability
Fireworks 3.

Usage
dom.pathIntersect()

Arguments
None.

Returns
Nothing.

Description
Performs an intersect operation on the selected paths.

dom.pathPunch()

Availability
Fireworks 3.

Usage
dom.pathPunch()

Arguments
None.

Returns
Nothing.

Description
Performs a punch operation on the selected paths.

dom.pathSimplify()

Availability
Fireworks 3.

Usage
dom.pathSimplify(limit)

Arguments
limit is a floating-point value that specifies how much to simplify. This value corresponds to the value in the Modify > Alter Path > Simplify dialog box.
Returns
Nothing.

Description
Performs a simplify operation on the selected paths.

dom.pathUnion()

Availability
Fireworks 3.

Usage
dom.pathUnion()

Arguments
None.

Returns
Nothing.

Description
Performs a union operation on the selected paths.

dom.placeTextInPath()

Availability
Fireworks 10.

Usage
dom.placeTextInPath()

Arguments
None.

Returns
Nothing.

Description
Places the selected text inside the selected path. If no text and path are selected, no action occurs.

dom.previewInBrowser()

Availability
Fireworks MX.
Usage
`dom.previewInBrowser(primaryBrowser)`

Arguments
`primaryBrowser`  A Boolean value that specifies which browser Fireworks should start: the primary browser (`true`) or the secondary browser (`false`).

Returns
Nothing.

Description
Previews the document in the primary or secondary browser.

`dom.previewAllPagesInBrowser()`

Availability
Fireworks 9.

Usage
`dom.previewAllPagesInBrowser()`

Arguments
Nothing.

Returns
Nothing.

Description
Previews the document in the primary browser.

`dom.rebuildColorTable()`

Availability
Fireworks 3.

Usage
`dom.rebuildColorTable()`

Arguments
None.

Returns
Nothing.
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.

dom.redefineStyleByName()

Availability
Fireworks 10.

Usage
dom.redefineStyleByName(sourcestyle, targetstyle)

Arguments
sourcestyle  Name of the source style.
targetstyle  Name of the style that is being redefined.

Returns
Nothing.

Description
Redefines the target style with the source style.

Example
The following command redefines the attributes of the target style "Chrome Reflective 016" with the attributes of the source style "Chrome Misc 011".

fw.getDocumentDOM().redefineStyleByName("Chrome Misc 011", "Chrome Reflective 016");

dom.redo()

Availability
Fireworks 3.

Usage
dom.redo()

Arguments
None.

Returns
Nothing.

Description
Reinstates the last action that was undone in the document.
dom.redraw()

Availability
Fireworks MX.

Usage
dom.redraw()

Arguments
None.

Returns
Nothing.

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

dom.redrawSmartShape()

Availability
Fireworks 8.

Usage
dom.redrawSmartShape()

Arguments
None.

Returns
Nothing.

Description
Forces the Auto Shapes in the document to redraw. This method is useful when modifying a Auto Shape outside a tool.

dom.reflectSelection()

Availability
Fireworks 3.

Usage
dom.reflectSelection(bHoriz, bVert, opts)

Arguments
bHoriz Boolean. If true, the items are reflected horizontally.
bVert  Boolean. If is true, the items are reflected vertically.

opts  Acceptable values are "transformAttributes", "autoTrimImages", and "autoTrimImages transformAttributes".

Returns
Nothing.

Description
Reflects the selection vertically, horizontally, or both.

**dom.removeAllGuides()**

Availability
Fireworks 3.

Usage
```javascript
dom.removeAllGuides(guidekind)
```

Arguments
```
guidekind  Acceptable values are "horizontal" and "vertical".
```

Returns
Nothing.

Description
Removes all guides of the specified type.

**dom.removeBehavior()**

Availability
Fireworks 3.

Usage
```javascript
dom.removeBehavior({event}, {eventIndex})
```

Arguments
```
event  An optional argument specifying the event that triggers the behavior. This argument is ignored by Fireworks.
eventIndex  An integer value that specifies the location of the behavior to be removed, starting with 0 (although, to specify the end location, pass –1 here). This argument is optional.
```

If you omit both optional arguments this function removes all events from selected hotspots and slices.

Returns
Nothing.
Description
Removes one or all behavior events from the selected hotspots and slices.

See also
dom.addBehavior()

dom.removeBrush()

Availability
Fireworks 3.

Usage
dom.removeBrush()

Arguments
None.

Returns
Nothing.

Description
Sets the brush of the selection to None.

dom.removeCharacterMarkup()

Availability
Fireworks 3.

Usage
dom.removeCharacterMarkup(tag)

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

Returns
Nothing.

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

dom.removeElementMask()

Availability
Fireworks 4.
Usage

`dom.removeElementMask(whatIfElementIsAnImage)`

Arguments

`whatIfElementIsAnImage` This argument is used only if the element (not the element mask) is an image. Acceptable values 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.

Description

Removes the mask from the selected element. If more than one element (or no elements) are selected when this function is called, Fireworks throws an exception.

`dom.removeFontMarkup()`

Availability

Fireworks 3.

Usage

`dom.removeFontMarkup(fontAttribute)`

Arguments

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

Returns

Nothing.

Description

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

`dom.removeFill()`

Availability

Fireworks 3.

Usage

`dom.removeFill()`

Arguments

None.

Returns

Nothing.
Description
Sets the fill of the selection to None.

`dom.removeGuide()`

Availability
Fireworks 3.

Usage
`dom.removeGuide(position, guidekind)`

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

Returns
Nothing.

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

`dom.removeNineScale()`

Availability
Fireworks CS3

Usage
`dom.removeNineScale(status)`

Arguments
None.

Returns
Nothing.

Description
Removes 9-slice scaling from the selected symbol.

Example
The following command removes 9-slice scaling from the selected symbol:

```javascript
fw.getDocumentDOM().removeNineScale();
```
**dom.removeTransformation()**

**Availability**
Fireworks 3.

**Usage**

```javascript
dom.removeTransformation()
```

**Arguments**
None.

**Returns**
Nothing.

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

**dom.reorderFrame()**

**Availability**
Fireworks 3.

**Usage**

```javascript
dom.reorderFrame(frameToMove, frameToPutItBefore, bMakeCopy)
```

**Arguments**

- `frameToMove` A zero-based index that specifies which frame to move or copy.
- `frameToPutItBefore` A zero-based index that specifies where to place the frame that is to be moved or copied. For example, if you pass 1 for `frameToMove` and 0 for `frameToPutItBefore`, the second frame is placed before the first frame.
- `bMakeCopy` Boolean. If `true`, the specified frame is copied instead of moved.

**Returns**
Nothing.

**Description**
Moves or copies the specified frame before another specified frame.

**Example**
The following command moves the third frame before the first frame:

```javascript
fw.getDocumentDOM().reorderFrame(2, 0, false);
```
**dom.reorderLayer()**

**Availability**
Fireworks 3, new arguments added in CS3.

**Usage**
```javascript
dom.reorderLayer(layerToMove, layerToPutItBefore, bMakeCopy, posInLayer, aboveBelowInto)
```

**Arguments**
- `layerToMove` A zero-based index that specifies which layer to move or copy.
- `layerToPutItBefore` A zero-based index that specifies where to place the layer to be moved or copied. For example, if you pass 1 for `layerToMove` and 0 for `layerToPutItBefore`, the second layer is placed before the first layer.
- `bMakeCopy` Boolean. If true, the specified layer is copied instead of moved.
- `aboveBelowInto` A zero-based index that specifies whether the layer being moved or copied will be a parent layer or a sub layer. A value of 0 indicates that the layer goes above the destination layer; a value of 1 indicates that the layer goes below the destination later, a value of 2 indicates that the layer goes into the destination layer to become a sub layer.

**Returns**
Nothing.

**Description**
Moves or copies the specified layer before another specified layer.

**dom.reorderPages()**

**Availability**
Fireworks CS3.

**Usage**
```javascript
dom.reorderPages(origPos, newPos)
```

**Arguments**
- `origPos` A long value that indicates the page number (position) of the page to be moved.
- `newPos` A long value that indicates the new position of the page within the document.

**Returns**
Nothing.

**Description**
Moves a specified page to a different location within the current document.
dom.replaceButtonTextStrings()

Availability
Fireworks 3.

Usage
dom.replaceButtonTextStrings(newString, uniformAttrs)

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

Returns
Nothing.

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.)

See also
dom.replaceButtonTextStringsInInstances()

dom.replaceButtonTextStringsInInstances()

Availability
Fireworks 3.

Usage
dom.replaceButtonTextStringsInInstances(newString, uniformAttrs)

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

Returns
Nothing.

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

dom.replaceTextString()

Availability
Fireworks 3.

Usage
dom.replaceTextString(newString, uniformAttrs)

Arguments
newString  Specifies the string to be used as replacement text.

uniformAttrs  Boolean. If false, each character retains the attributes of the character that was formerly in its position; that is, Fireworks preserves the existing formatting. If true, all characters assume the attributes of the first character in the string that is being replaced.

Returns
Nothing.

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

dom.resetNineScale()

Availability
Fireworks CS3

Usage
dom.resetNineScale()

Arguments
None.

Returns
Nothing.

Description
Resets 9-slice scaling for the selected symbol back to default.

Example
The following command resets 9-slice scaling for the selected symbol:

fw.getDocumentDOM().resetNineScale();
**dom.resizeSelection()**

Availability
Fireworks 3.

Usage
`dom.resizeSelection(width, height)`

Arguments
- `width` and `height` Integers that specify the new width and height, in pixels.

Returns
Nothing.

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

**dom.restoreJPEGMask()**

Availability
Fireworks 4.

Usage
`dom.restoreJPEGMask()`

Arguments
None.

Returns
Nothing.

Description
Restores the selection that is specified in `dom.saveJPEGMask()`.

See also
`dom.saveJPEGMask()`

**dom.restoreSelection()**

Availability
Fireworks 4.

Usage
`dom.restoreSelection(selectionName, fromDocument, {operation}, {invert})`
Arguments

**selectionName**  User-specified name of the selection to restore. If selection name is not specified, the selection named "default" will be restored.

**fromDocument**  Index of a currently open document from which to load the selection. If the from document is not specified, the selection will be restored from the active document.

**operation**  Operation to perform on the selection and on the document being loaded. Acceptable values are "new or replace", "add", "subtract", and "intersect". This parameter is optional. If the operation is not specified, Fireworks will behave as if "new or replace" is specified.

**invert**  A Boolean value that determines if the selection should be inverted before performing the operation on it. This parameter is optional. If invert is not specified, defaults to false.

Returns

true if the selection is inverted; otherwise false.

Description

Restores the selection that is specified in dom.saveSelection().

See also

dom.saveSelection()

dom.reversePathTextDirection()  

**Availability**
Fireworks 3.

**Usage**

dom.reversePathTextDirection()

Arguments

None.

Returns

Nothing.

Description

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

**dom.rotateDocument()**

**Availability**
Fireworks 3.

**Usage**

dom.rotateDocument(rotationAmount)
Arguments
rotationAmount Acceptable values for rotationAmount are 90, 180, and 270.

Returns
Nothing.

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

dom.rotateSelection()

Availability
Fireworks 3.

Usage
dom.rotateSelection(rotationDegrees, opts)

Arguments
rotationDegrees A floating-point value that specifies the number of degrees to rotate the selection.

opts Acceptable values are "transformAttributes", "autoTrimImages", and "autoTrimImages transformAttributes".

Returns
Nothing.

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

dom.save()

Availability
Fireworks 3.

Usage
dom.save({bOkToSaveAs})

Arguments
bOkToSaveAs If this optional argument 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
ture if the save operation is successful; false otherwise.
Description
Saves the document in its default location. After a successful save operation, the document’s `isDirty` property is cleared.

**dom.saveCopyAs()**

Availability
Fireworks 3.

Usage
`dom.saveCopyAs(fileURL)`

Arguments
- **fileURL**: 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 is successful; `false` otherwise.

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.

**dom.saveJPEGMask()**

Availability
Fireworks 4.

Usage
`dom.saveJPEGMask()`

Arguments
None.

Returns
Nothing.

Description
Stores the current selection in bitmap mode as the “Selective JPEG mask”. Use `dom.restoreJPEGMask()` to restore the JPEG mask.

See also
- `dom.restoreJPEGMask()`
**dom.saveSelection()**

**Availability**
Fireworks 4.

**Usage**

```
dom.saveSelection({selectionName}, {toDocument}, {operation})
```

**Arguments**

- **selectionName** User-specified name of the selection to save. This parameter is optional. If `selectionName` is not specified, Fireworks will save the selection with the name "default".
- **toDocument** Index of a currently open document where the selection will be saved. This parameter is optional. If `toDocument` is not specified, Fireworks will save the selection to the active document.
- **operation** The operation to be performed on the selection and on the selection being loaded from the `selectionName` parameter. Acceptable values are "new or replace", "add", "subtract", and "intersect". This parameter is optional. If `operation` is not specified, Fireworks treats it as if it were specified as "new or replace".

**Returns**
Nothing.

**Description**
Stores the current selection in bitmap mode as the saved selection. Use `dom.restoreSelection()` to restore the selection.

**See also**
`dom.restoreSelection()`

**dom.scaleSelection()**

**Availability**
Fireworks 3.

**Usage**

```
dom.scaleSelection(xScaleAmount, yScaleAmount, opts)
```

**Arguments**

- **xScaleAmount** and **yScaleAmount**  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%, 2 represents 200%, and so on.
- **opts**  Acceptable values are "transformAttributes", "autoTrimImages", and "autoTrimImages transformAttributes".

**Returns**
Nothing.
Description
Scales the selection in the horizontal and vertical axes.

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

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

**dom.scalingGridRect()**

Availability
Fireworks 9.

Usage
`dom.scalingGridRect(left, top, right, bottom)`

Arguments
- `bounds` Specifies the bounds of the grid rectangle.

Returns
Nothing.

Description
Sets the positions of the 9-slice scaling guides by specifying the size of the rectangle they surround.

Example
The following command sets the 9-slice scaling grid rectangle to -50, -50, 50, 50:

```javascript
fw.getDocumentDOM().scalingGridRect({-50, -50, 50, 50});
```

**dom.selectAdjustPixelSel()**

Availability
Fireworks 3.

Usage
`dom.selectAdjustPixelSel(whatToDo, amount)`

Arguments
- `whatToDo` Acceptable values are "expand", "contract", "border", and "smooth".
  - 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`.
- `amount` An integer specifying the amount by which to adjust. Any integer is acceptable.
Returns
Nothing.

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.

\textbf{dom.selectAll()}

Availability
Fireworks 3.

Usage
\texttt{dom.selectAll()}

Arguments
None.

Returns
Nothing.

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.

\textbf{dom.selectAllOnLayer()}

Availability
Fireworks MX.

Usage
\texttt{dom.selectAllOnLayer(\textit{layerIndex}, \textit{bRememberSelection}, \textit{bToggleSelection})}

Arguments
\textit{layerIndex} A long integer that identifies the layer on which to select the element.
\textit{bRememberSelection} A Boolean value. If \textit{true}, all the elements on the layer are appended to the current selection.
\textit{bToggleSelection} A Boolean value. Toggles the selection of elements instead of simply selecting them. This parameter is useful only if \textit{bRememberSelection} is \textit{true}.

Returns
Nothing.

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.
dom.selectChildren()

Availability
Fireworks 3.

Usage
dom.selectChildren()

Arguments
None.

Returns
Nothing.

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.

See also
dom.selectParents()

dom.selectFeather()

Availability
Fireworks 3.

Usage
dom.selectFeather(featherAmount)

Arguments
featherAmount An integer that specifies the number of pixels by which to feather the selection.

Returns
Nothing.

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

dom.selectInverse()

Availability
Fireworks 3.

Usage
dom.selectInverse()
Arguments
None.

Returns
Nothing.

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

`dom.selectNone()`

Availability
Fireworks 3.

Usage
`dom.selectNone()`

Arguments
None.

Returns
Nothing.

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.

`dom.selectParents()`

Availability
Fireworks 3.

Usage
`dom.selectParents()`

Arguments
None.

Returns
Nothing.

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.
See also
dom.selectChildren()

**dom.selectSimilar()**

**Availability**
Fireworks 3.

**Usage**
dom.selectSimilar(tolerance, edgemode, featherAmt, combinemode)

**Arguments**
- **tolerance** An integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- **edgemode** Acceptable values are "hard edge", "antialias", and "feather".
- **featherAmt** 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.

**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.

See also
dom.selectSimilarFromPoint()

**dom.selectSimilarFromPoint()**

**Availability**
Fireworks 3.

**Usage**
dom.selectSimilarFromPoint(where, tolerance, edgemode, featherAmt, combinemode)

**Arguments**
- **where** A point that specifies the x,y coordinates of the pixel whose color is used to calculate the new mask (see "Point data type" on page 9).
- **tolerance** An integer between 0 and 255, inclusive, that specifies the tolerance for selecting pixels.
- **edgemode** Acceptable values are "hard edge", "antialias", and "feather".
- **featherAmt** 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.

**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.

**See also**
`dom.selectSimilar()`

**dom.sendEmail()**

**Availability**
Fireworks MX 2004.

**Usage**
`dom.sendEmail(fileAttachment)`

**Arguments**
- `fileAttachment` A string, which is expressed as `file://URL`, denoting the location of a file to send by e-mail.

**Returns**
Nothing.

**Description**
Creates a new e-mail with the specified file as an attachment.

**Example**
The following example opens a new e-mail in the default e-mail program and attaches the file foo.png to the message:

```javascript
fw.getDocumentDOM().sendEmail("file://Users/andy/Documents/foo.png");
```

**dom.setAllLayersDisclosure()**

**Availability**
Fireworks 4.

**Usage**
`dom.setAllLayersDisclosure(bDisclosed)`

**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.

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

See also
dom.setLayerDisclosure()

dom.setAnimInstanceLoopCount()

Availability
Fireworks 3, deprecated in 4 in favor of dom.setAnimInstanceNumFrames().

Usage
dom.setAnimInstanceLoopCount(\texttt{loopCount})

Arguments
\texttt{loopCount} An integer that corresponds to the loop count value that appears in the Objects panel when a multi-frame image instance is selected.

Returns
Nothing.

Description
Sets the loop count of the selected instances of multi-frame image symbols.

dom.setAnimInstanceNumFrames()

Availability
Fireworks 4.

Usage
dom.setAnimInstanceNumFrames(\texttt{numFrames})

Arguments
\texttt{numFrames} An integer that specifies the number of frames through which the symbol animates.

Returns
Nothing.

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

See also
dom.convertToAnimSymbol()
**dom.setAnimInstanceOffsetDist()**

**Availability**
Fireworks 4.

**Usage**

```javascript
dom.setAnimInstanceOffsetDist(offsetDistPt)
```

**Arguments**

- `offsetDistPt` 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 25 pixels down.

**Returns**

Nothing.

**Description**

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

**See also**

`dom.convertToAnimSymbol()`

---

**dom.setAnimInstanceRotationAmount()**

**Availability**
Fireworks 4.

**Usage**

```javascript
dom.setAnimInstanceRotationAmount(rotationAmount)
```

**Arguments**

- `rotationAmount` A floating-point 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.

**Description**

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

**See also**

`dom.convertToAnimSymbol()`
**dom.setAnimInstanceScaleAmount()**

Availability
Fireworks 4.

Usage
```
dom.setAnimInstanceScaleAmount(scaleAmount)
```

Arguments
- **scaleAmount** A positive floating-point value that specifies the amount of scaling to be applied to the animation symbol. For example, pass 50 to scale the symbol to 50% of its current size, and pass 200 to scale it to twice its current size. To specify no scaling, pass 100.

Returns
Nothing.

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

See also
- `dom.convertToAnimSymbol()`

**dom.setAnimInstanceStartEndOpacity()**

Availability
Fireworks 4.

Usage
```
dom.setAnimInstanceStartEndOpacity(startOpacity, endOpacity)
```

Arguments
- **startOpacity** and **endOpacity** Float values between 0 and 100 that specify the starting and ending opacity of the animation symbol.

Returns
Nothing.

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

See also
- `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.

Usage
dom.setAnimInstanceStartFrame(startFrame)

Arguments
startFrame An integer that corresponds to the starting frame value that appears in the Objects panel when a multi-frame image instance is selected.

Returns
Nothing.

Description
Sets the start frame of the selected instances of multi-frame image symbols.

dom.setBlendMode()

Availability
Fireworks 3.

Usage
dom.setBlendMode(mode)

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

Returns
Nothing.

Description
Specifies the blend mode of the selection.

dom.setBrush()

Availability
Fireworks 3.

Usage
dom.setBrush(brush)

Arguments
brush A Brush object (see “Brush object” on page 243).
Returns
Nothing.

Description
Sets the selection to the specified brush.

See also
dom.setBrushColor(), dom.setBrushName(), dom.setBrushNColorNTextrue(), dom.setBrushPlacement()

dom.setBrushColor()

Availability
Fireworks 3.

Usage
dom.setBrushColor(color)

Arguments
color A color string (see “Color string data type” on page 8).

Returns
Nothing.

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

See also
dom.setBrushNColorNTextrue()

dom.setBrushName()

Availability
Fireworks 3.

Usage
dom.setBrushName(category, currentName, newName)

Arguments
category A string that specifies the category of the brush to be renamed.
currentName A string that specifies the current name of the brush.
newName A string that specifies the new name of the brush.

Returns
Nothing.
Description
Renames a brush. Does not change the brush category.

dom.setBrushNColor()

Availability
Fireworks 3.

Usage
dom.setBrushNColor(brush, brushColor)

Arguments
brush A Brush object (see “Brush object” on page 243).
brushColor A color string (see “Color string data type” on page 8).

Returns
Nothing.

Description
Sets the selection to the specified brush and brush color.

dom.setBrushNColorNTexture()

Availability
Fireworks 3.

Usage
dom.setBrushNColorNTexture(brush, color, texture-name)

Arguments
brush A Brush object (see “Brush object” on page 243).
color A color string (see “Color string data type” on page 8).
texture-name The name of the texture to be applied.

Returns
Nothing.

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

See also
dom.setBrushColor()
Extending Fireworks

The Document object

**dom.setBrushPlacement()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.setBrushPlacement(placement)
```

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

**Returns**
Nothing.

**Description**
Specifies the brush placement of the stroke on the selection.

**dom.setButtonAutoSlice()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.setButtonAutoSlice(bAutoSlice)
```

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

**Returns**
Nothing.

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

**dom.setButtonIncludeDownState()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.setButtonIncludeDownState(bIncludeDownState)
```

**Arguments**
bIncludeDownState Boolean. If true, the Down state is included in the button; if false, it is not.
Returns
Nothing.

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

**dom.setButtonIncludeOverWhileDownState()**

Availability
Fireworks 3.

Usage
`dom.setButtonIncludeOverWhileDownState(bIncludeOverWhileDownState)`

Arguments
- `bIncludeOverWhileDownState`  Boolean. If `true`, the Over-While-Down state is included in the button; if `false`, it is not.

Returns
Nothing.

Description
If the user edits a Button document, this function specifies whether to include the Over-While-Down state in a button.

**dom.setButtonShowDownOnLoad()**

Availability
Fireworks 3.

Usage
`dom.setButtonShowDownOnLoad(bShowDownOnLoad)`

Arguments
- `bShowDownOnLoad`  Boolean. If `true`, the Down-State-on-Load is shown in the button; if `false`, it is not.

Returns
Nothing.

Description
If the user edits a Button document, this function specifies whether to show the Down-State-on-Load in a button.

**dom.setButtonOptions()**

Availability
Fireworks 3.
Usage

```
dom.setButtonOptions(exportOptions, URLString, altTagString, targetTagString, sliceName, statusMessage)
```

Arguments

- **exportOptions** An ExportOptions object (see “ExportOptions object” on page 260).
- **URLString** 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** 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** 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.

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.

**dom.setDefaultBrushAndFillColors()**

Availability

Fireworks 3.

Usage

```
dom.setDefaultBrushAndFillColors()
```

Arguments

None.

Returns

Nothing.

Description

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

**dom.setDefaultFillVector()**

Availability

Fireworks 3.

Usage

```
dom.setDefaultFillVector()
```
Arguments
None.

Returns
Nothing.

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

```javascript
dom.setDocumentCanvasColor()
```

Availability
Fireworks 3.

Usage
```javascript
dom.setDocumentCanvasColor(color)
```

Arguments
color A color string (see “Color string data type” on page 8).

Returns
Nothing.

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

Example
The following command sets the canvas color to blue:

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

```javascript
dom.setDocumentCanvasSize()
```

Availability
Fireworks 3, with new argument added in Fireworks CS3.

Usage
```javascript
dom.setDocumentCanvasSize(boundingRectangle, currentPageOnly)
```

Arguments
boundingRectangle A rectangle that specifies the new canvas size for the document, in pixels (see “Rectangle data type” on page 9). Any items outside the specified rectangle are removed.

currentPageOnly A Boolean value that specifies whether the change in canvas size applies to all pages or only the current page. If the value is true, only the current page is resized. If it is false, all pages of the document are resized. The default value is true.
Returns
Nothing.

Description
Sets the document’s canvas size to the specified rectangle and apply the change to the current page or all pages.

Example
The following command sets the canvas to a size of 200 by 200 pixels:
```
```

**dom.setDocumentCanvasSizeToDocumentExtents()**

Availability
Fireworks 3.

Usage
```
dom.setDocumentCanvasSizeToDocumentExtents(bGrowCanvas)
```

Arguments
- **bGrowCanvas**: Boolean. If `true`, the canvas can expand or shrink in size; if `false`, it only shrinks.

Returns
Nothing.

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.

Example
The following command resizes the canvas to include all the items in the document, enlarging the canvas if necessary:
```
fw.getDocumentDOM().setDocumentCanvasSizeToDocumentExtents(true);
```

See also
“**dom.setDocumentCanvasSizeToSelection()**” on page 146

**dom.setDocumentCanvasSizeToSelection()**

Availability
Fireworks 3.

Usage
```
dom.setDocumentCanvasSizeToSelection()
```

Arguments
None.
Returns
Nothing.

Description
Calculates the size of all the items in the selection and resizes the document canvas accordingly.

See also
dom.setDocumentCanvasSizeToDocumentExtents()

dom.setDocumentImageSize()

Availability
Fireworks 3, with additional argument added in Fireworks CS3.

Usage
dom.setDocumentImageSize(boundingRectangle, resolution, currentPageOnly)

Arguments
boundingRectangle A rectangle that specifies the size to which the document should be scaled (see “Rectangle data type” on page 9).
resolution Specifies the resolution for the scaled document (see “Resolution data type” on page 9).
currentPageOnly A Boolean value that specifies whether the change in document size applies to all pages or only the current page. If the value is true, only the current page is resized. If it is false, all pages of the document are resized. The default value is true.

Returns
Nothing.

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

dom.setDocumentResolution()

Availability
Fireworks 3.

Usage
dom.setDocumentResolution(resolution)

Arguments
resolution Specifies the resolution for the document (see “Resolution data type” on page 9).

Returns
Nothing.
Description
Sets the resolution of the document.

dom.setEffectName()

Availability
Fireworks MX.

Usage
dom.setEffectName(category, oldName, newName)

Arguments
category A string that defines the name of the category of the effect.
oldName The existing name of the effect.
newName The new name to give to the effect.

Returns
Nothing.

Description
Sets the name for the current effect.

dom.setElementLocked()

Availability
Fireworks 8.

Usage
dom.setElementLocked(frameNum, layerNum, objectIndex, bBlock, bAllLayers, bLockLayers)

Arguments
frameNum A zero-based integer that specifies the frame that contains the element or elements to be locked. To specify the current frame, pass -1.
layerNum A zero-based integer that specifies the layer that contains the element or elements to be locked. To specify the current layer, pass -1.
objectIndex A zero-based integer that specifies the element or elements to lock or unlock. 0 represents the topmost element in the specified layer. To lock or unlock all the elements in the specified layer, pass -1.
bBlock A Boolean value. If true, the element or elements are to be locked; if false, the elements are unlocked.
bAllLayers A Boolean value. If true, all layers are specified; false otherwise.
bLockLayers A Boolean value. If true, locks all layers; false otherwise.

Returns
Nothing.
**Description**
Sets the name of the selected element or elements.

**dom.setElementLockedByName()**

**Availability**
Fireworks 8.

**Usage**
dom.setElementLockedByName(name, bLock)

**Arguments**
- name A string that specifies the name of the element or elements to be locked or unlocked. If more than one element has the same name, the function locks or unlocks all of them.
- bBlock A Boolean value. If true, the element or elements are locked; if false, they are unlocked.

**Returns**
An array of the elements for which the lock status is set.

**Description**
Locks or unlocks all the elements with the specified name. If no element has the specified name an exception is thrown. If elements are hidden (for example, if they are on a hidden layer or frame), the function will not lock them.

**dom.setElementMaskMode()**

**Availability**
Fireworks 4.

**Usage**
dom.setElementMaskMode(mode)

**Arguments**
- mode Acceptable values are "mask to image" and "mask to path".

**Returns**
Nothing.

**Description**
Sets the rendering mode on the selected element’s element mask. Only one element can be selected when calling this function. If more than one element (or no elements) are selected when this function is called, Fireworks throws an exception. Fireworks also returns an error if the selected element has no element mask.
**dom.setElementMaskShowAttrs()**

**Availability**
Fireworks 4.

**Usage**
dom.setElementMaskShowAttrs(bShow)

**Arguments**
bShow  Boolean. If true, the vector mask fill and stroke are visible; if false, they are hidden.

**Returns**
Nothing.

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

**dom.setElementName()**

**Availability**
Fireworks 3.

**Usage**
dom.setElementName(name)

**Arguments**
name  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.

**Description**
Sets the name of the selected element(s).

**See also**

dom.findNamedElements()

**dom.setElementVisible()**

**Availability**
Fireworks 4.

**Usage**
dom.setElementVisible(frameIndex, layerIndex, elementIndex, bShow)
Arguments

frameIndex  An integer value that specifies the frame that contains the element(s) to be shown or hidden, starting with 0 (although, to specify the current frame, pass –1 here).

layerIndex  An integer value that specifies the layer that contains the element(s) to be shown or hidden, starting with 0 (although, to specify the current layer, pass –1 here).

elementIndex  An integer value that specifies the element(s) to show or hide, starting with 0 (although, to show or hide all the elements in the specified layer, pass –1 here).

bShow  Boolean. If true, the element(s) are visible; they are hidden otherwise.

Returns

Nothing.

Description

Shows or hides the specified element(s).

Example

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

\[\text{fw.getDocumentDOM().setElementVisible(-1, -1, -1, false)}\]

See also

dom.setElementVisibleByName()
**dom.setExportOptions()**

Availability
Fireworks 3.

Usage
`dom.setExportOptions(exportOptions)`

Arguments
- `exportOptions` An ExportOptions object (see “ExportOptions object” on page 260).

Returns
Nothing.

Description
Sets the document Export Options.

**dom.setExportSettings()**

Availability
Fireworks 3.

Usage
`dom.setExportSettings(exportSettings)`

Arguments
- `exportSettings` An ExportSettings object (see “ExportSettings object” on page 263).

Returns
Nothing.

Description
Sets the document export settings.

**dom.setFill()**

Availability
Fireworks 3.

Usage
`dom.setFill(fill)`

Arguments
- `fill` A Fill object (see “Fill object” on page 265).
DOM.setFillColor()

Availability
Fireworks 3.

Usage
DOM.setFillColor(color)

Arguments

- color A color string (see “Color string data type” on page 8).

Returns
Nothing.

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

DOM.setFillEdgeMode()

Availability
Fireworks 3.

Usage
DOM.setFillEdgeMode(edgemode, featherAmt)

Arguments

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

Returns
Nothing.

Description
Sets the edge type for selected items with fills.
**dom.setFillNColor()**

**Availability**
Fireworks MX.

**Usage**
dom.setFillNColor(fill, color)

**Arguments**
fill A Fill object (see "Fill object" on page 265).
color A color string (see "Color string data type" on page 8).

**Returns**
Nothing.

**Description**
Sets the selection to the specified fill and fill color.

**dom.setFillNColorNTexture()**

**Availability**
Fireworks 3.

**Usage**
dom.setFillNColorNTexture(fill, color, texture-name)

**Arguments**
fill A Fill object (see "Fill object" on page 265).
color A color string (see "Color string data type" on page 8).
texture-name The name of the texture to be applied.

**Returns**
Nothing.

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

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

```javascript
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.

**Usage**
`dom.setFillPlacement(placement)`

**Arguments**
`placement` Acceptable values are "top" and "bottom".

**Returns**
Nothing.

**Description**
Sets the fill placement for selected items with fills.

**dom.setFillVector()**

**Availability**
Fireworks 3.

**Usage**
`dom.setFillVector(p1, p2, p3)`

**Arguments**
`p1`, `p2`, and `p3` Points that specify the x,y coordinates of the three points used to calculate the fill vector (see “Point data type” on page 9).

**Returns**
Nothing.

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

**dom.setFillVectorStart()**

**Availability**
Fireworks 3.

**Usage**
`dom.setFillVectorStart(p1)`

**Arguments**
`p1` A point that specifies the x,y coordinates of the fill start point (see “Point data type” on page 9).
Returns
Nothing.

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.

`dom.setGradientName()`

Availability
Fireworks 3.

Usage
`dom.setGradientName(currentName, newName)`

Arguments

- **currentName** A string that specifies the current name of the gradient.
- **newName** A string that specifies the new name of the gradient.

Returns
Nothing.

Description
Renames a gradient.

`dom.setGridOrigin()`

Availability
Fireworks 3.

Usage
`dom.setGridOrigin(gridOrigin)`

Arguments

- **gridOrigin** A point that specifies the x,y coordinates of the document’s grid origin (see “Point data type” on page 9).

Returns
Nothing.

Description
Sets the grid origin for the document.


**dom.setSizeGrid()**

**Availability**
Fireworks 3.

**Usage**

```
dom.setSizeGrid(gridSize)
```

**Arguments**

- **gridSize**
  A point that specifies the x,y coordinates that are used for the document’s grid size (see “Point data type” on page 9).

**Returns**

Nothing.

**Description**

Sets the grid size for the document.

**dom.setColorGrid()**

**Availability**
Fireworks 3.

**Usage**

```
dom.setColorGrid(gridColor)
```

**Arguments**

- **gridColor**
  A color string (see “Color string data type” on page 8).

**Returns**

Nothing.

**Description**

Sets the color used to display the grid.

**dom.setGroupType()**

**Availability**
Fireworks 3, arguments deprecated in Fireworks 4.

**Usage**

```
dom.setGroupType({type})
```

Arguments
type 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. (The "mask to image" and "mask to path" values are deprecated in Fireworks 4.)

Returns
Nothing.

Description
Changes the group type of the currently selected groups.

dom.setGuideColor()

Availability
Fireworks 3.

Usage
dom.setGuideColor(guideColor)

Arguments
guideColor A color string (see “Color string data type” on page 8).

Returns
Nothing.

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

See also
dom.setSliceGuideColor()

dom.setHotspotAltTag()

Availability
Fireworks 3.

Usage
dom.setHotspotAltTag(whatToSet, altTagString)

Arguments
whatToSet Acceptable values are "hotspots", "slices", and "hotspots and slices".
altTagString A string that specifies the text to be used for the alt tag.

Returns
Nothing.
Description
Sets the alt tag text to the specified value for the hotspots and slices in the selection.

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.

Usage
`dom.setHotSpotColor(whatToSet, color)`

Arguments
- `whatToSet` Acceptable values are "hotspots", "slices", and "hotspots and slices".
- `color` A color string (see "Color string data type" on page 8).

Returns
Nothing.

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

Example
The following command sets the color of the selected hotspots to red:
fw.getDocumentDOM().setHotspotColor("hotspots", "#ff0000");

`dom.setHotspotRectangle()`

Availability
Fireworks 3.

Usage
`dom.setHotSpotRectangle(boundingRectangle, bMakeCopy)`

Arguments
- `boundingRectangle` A rectangle that specifies the size of the new hotspot or slice (see "Rectangle data type" on page 9).
- `bMakeCopy` A Boolean value; if it is `true`, the selection is copied and resized instead of moved and resized.

Returns
Nothing.
Description
If the selection is a single hotspot or slice, this function moves or copies it to the specified location at the specified size.

`dom.setHotspotShape()`

Availability
Fireworks 3.

Usage
`dom.setHotspotShape(whatToSet, shape)`

Arguments
- `whatToSet` Acceptable values are "hotspots", "slices", or "hotspots and slices".
- `shape` Acceptable values are "rectangle", "oval", or "polyline".

Returns
Nothing.

Description
Sets the specified hotspots and slices in the selection to the specified shape.

`dom.setHotspotTarget()`

Availability
Fireworks 3.

Usage
`dom.setHotspotTarget(whatToSet, targetTagString)`

Arguments
- `whatToSet` Acceptable values are "hotspots", "slices", or "hotspots and slices".
- `targetTagString` A string that specifies the text to be used for the target tag.

Returns
Nothing.

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

Example
The following command links the currently selected slices to the parent window:

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

Availability
Fireworks 3.

Usage
dom.setHotspotText(whatToSet, textString, urlToMatch, bUpdateAttributes)

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

Arguments
whatToSet Acceptable values are "hotspots", "slices", or "hotspots and slices".
textString A string that specifies the text to be used for the hotspot or slice.
urlToMatch 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.

bUpdateAttributes Boolean. If 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.

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

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.

Usage
dom.setLayerDisclosure(layerIndex, bDisclosed)
Arguments

layerIndex  An integer value that specifies the layer that contains the elements to be displayed or hidden, starting with 0 (although, to specify the current layer, pass –1 here).

bDisclosed  Boolean. If true, all elements on the specified layer are displayed in the Layers list; if false, only the layer name appears on the list.

Returns

Nothing.

Description

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

See also

dom.setAllLayersDisclosure()

**dom.setLayerLocked()**

Availability

Fireworks 3.

Usage

dom.setLayerLocked(layerIndex, frameIndex, bLock, bAllLayers)

Arguments

layerIndex  An integer value that specifies the layer to be locked or unlocked, starting with 0 (although, to specify the current layer, pass –1 here). To lock or unlock all the layers on a frame, use the bAllLayers argument.

frameIndex  An integer value that specifies the frame that contains the layer that is to be locked or unlocked, starting with 0 (although, to specify the current frame, pass –1 here).

bLock  Boolean. If true, the layer is locked; if false, the layer is unlocked.

bAllLayers  Boolean. If true, all the layers on the specified frame are locked or unlocked, and any value passed for layerIndex is ignored.

Returns

Nothing.

Description

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

Example

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

```javascript
da.setLayerLocked(1, 0, true, true);
```
dom.setLayerName()

Availability
Fireworks 3.

Usage
```
dom.setLayerName(layerIndex, layerName)
```

Arguments
- `layerIndex`: An integer value that specifies the layer to be renamed, starting with 0 (although, to specify the current layer, pass -1 here).
- `layerName`: A string that specifies the new name for the layer.

Returns
Nothing.

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

dom.setLayerSharing()

Availability
Fireworks 3.

Usage
```
dom.setLayerSharing(layerIndex, sharedStatus, bUnshareCopiesToAllFrames, bWarnUser)
```

Arguments
- `layerIndex`: An integer value that specifies the layer to be shared or not shared, starting with 0 (although, to specify the current layer, pass -1 here).
- `sharedStatus`: Acceptable values are "shared" or "not shared".
- `bUnshareCopiesToAllFrames`: A Boolean value 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.
- `bWarnUser`: Boolean. 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.

Description
Changes the Shared layer status of a layer.
Example
The following command sets the selected layer to Shared and displays a warning that data loss is possible:

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

**dom.setLayerVisible()**

**Availability**
Fireworks 3.

**Usage**

```javascript
dom.setLayerVisible(layerIndex, frameIndex, bShow, bAllLayers)
```

**Arguments**

- `layerIndex` An integer value that specifies the layer that should be shown or hidden, starting with 0 (although, to specify the current layer, pass -1 here). To show or hide all the layers on a frame, use the `bAllLayers` argument.

- `frameIndex` An integer value that specifies the frame that contains the layer to be shown or hidden, starting with 0 (although, to specify the current frame, pass -1 here). A zero-based integer specifying the frame that contains the layer to be shown or hidden.

- `bShow` Boolean. If true, the layer is visible; if false, the layer is hidden.

- `bAllLayers` Boolean. If true, all the layers on the specified frame are shown or hidden, and any value that is passed for `layerIndex` is ignored.

**Returns**
Nothing.

**Description**
Shows or hides a layer on the specified frame.

**dom.setMasterPage()**

**Availability**
Fireworks CS3.

**Usage**

```javascript
dom.setMasterPage(PageNum)
```

**Arguments**

- `PageNum` An long value that specifies the zero-based index of the page number to be set as the master page.

**Returns**
Nothing.

**Description**
Sets the specified page to be the document’s master page. For example:
**dom.setMatteColor()**

**Availability**
Fireworks 3.

**Usage**
dom.setMatteColor(bUseMatteColor, matteColor)

**Arguments**

- **bUseMatteColor**  Boolean. If true, the document’s matte color is set to the value that is specified by matteColor. If false, any matte color is removed from the document, and the second argument is ignored.
- **matteColor**  A color string (see “Color string data type” on page 8).

**Returns**
Nothing.

**Description**
Sets or removes the document’s matte color that is used for exporting.

**Example**
The following command sets the matte color to blue:
fw.getDocumentDOM().setMatteColor(true, "#0033ff");

**dom.setNineScaleGuideColor()**

**Availability**
Fireworks 9.

**Usage**
dom.setNineScaleGuideColor(color)

**Arguments**

- **color**  Color string as RGB value.

**Returns**
Nothing.

**Description**
Sets the color that is used to display 9-slice (nonslice) guides.
**dom.setPixelMask()**

**Availability**
Fireworks 3, deprecated in 4 in favor of `dom.setSelectionMask()`.

**Usage**
```javascript
dom.setPixelMask(mask, howToCombineMasks)
```

**Arguments**
- `mask` A mask variable that specifies the mask to be applied (see "Mask data type" on page 8). If `mask` is `null`, any existing pixel-selection mask is removed.
- `howToCombineMasks` 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.

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

**See also**
`dom.setSelectionMask()`

**dom.setOnionSkinning()**

**Availability**
Fireworks 3.

**Usage**
```javascript
dom.setOnionSkinning(before, after)
```

**Arguments**
- `before` and `after` Integers that specify the number of frames to display before and after the current one. To disable onion skinning, pass 0 for both arguments. To enable onion skinning for all frames, pass 0 for `before` and a large number (for example, 99,999) for `after`.

**Returns**
Nothing.

**Description**
Sets the onion-skinning options for the document.

**Example**
The following command turns on onion skinning two frames before the selected frame and zero frames after it:
```javascript
fw.getDocumentDOM().setOnionSkinning(2, 0);
```
The Document object

**dom.setOpacity()**

**Availability**
Fireworks 3.

**Usage**
```
dom.setOpacity(opacity)
```

**Arguments**
- `opacity` A float variable between 0 and 100, inclusive.

**Returns**
Nothing.

**Description**
Sets the opacity of the selection to the specified value.

**Example**
The following command sets the selected item to an opacity of 55%:
```
fw.getDocumentDOM().setOpacity(55);
```

**dom.setPageName()**

**Availability**
Fireworks CS3.

**Usage**
```
dom.setPageName(index, name)
```

**Arguments**
- `index` An long value that specifies the page number of the page to be renamed.
- `name` A string that specifies the new name for the page.

**Returns**
Nothing.

**Description**
Renames a page. For example:
```
fw.getDocumentDOM().setPageName(0, "new name")
```

**dom.setQuadrangle()**

**Availability**
Fireworks 3.
Usage

dom.setQuadrangle(pTopLeft, pTopRight, pBottomRight, pBottomLeft, options)

Arguments

pTopLeft, pTopRight, pBottomRight, and pBottomLeft  Relative coordinates of a quadrangle expressed as percentages of an arbitrary square. These are not specific x, y coordinates.

options  Acceptable values are "transformAttributes", "autoTrimImages", and "autoTrimImages transformAttributes".

Returns

Nothing.

Description

Creates a bounding quadrangle based on percentages of an arbitrary square, and then transforms the selection within the 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.

Example

The following command performs the transform operation on the selection within the specified points:

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; modified in Fireworks 10.

Usage

dom.setRectRoundness(roundness, mode)

Arguments

roundness  A floating-point value between 0 and 1 that specifies the roundness to use for the corners (0 is no roundness, 1 is 100% roundness).

mode  String that specifies the mode of corner roundness of the rectangle as either a percentage value or exact pixel value.

Returns

Nothing.

Description

Modifies the corner roundness of all the selected rectangle primitives.

See also

dom.addNewRectanglePrimitive(), dom.setRectSides()
**dom.setRectRoundnessMode()**

**Availability**
Fireworks 4.

**Usage**

```javascript
dom.setRectRoundnessMode(mode)
```

**Arguments**

- `mode`  
  String that specifies the mode of corner roundness of the rectangle as either a percentage value or exact pixel value.

**Returns**

Nothing.

**Description**

Specifies the mode of corner roundness of the rectangle as either a percentage value or exact pixel value.

**See also**

`dom.addNewRectanglePrimitive()`, `dom.setRectSides()`

**dom.setRectSides()**

**Availability**
Fireworks 4.

**Usage**

```javascript
dom.setRectSides(newSides)
```

**Arguments**

- `newSides`  
  A rectangle that specifies the new untransformed sides of the rectangle primitive (see “Rectangle data type” on page 9). Rectangle primitives remember their transformations, so the user sees the transformed result of `newSides` in the document.

**Returns**

Nothing.

**Description**

Modifies the untransformed sides of all selected rectangle primitives.

**See also**

`dom.addNewRectanglePrimitive()`, `dom.setRectRoundnessMode()`
**dom.setSelectionBounds()**

**Availability**
Fireworks 3.

**Usage**
dom.setSelectionBounds(boundingRectangle, opts)

**Arguments**
boundingRectangle  A rectangle that specifies the new location and size of the selection (see “Rectangle data type” on page 9).

opts  Acceptable values are "transformAttributes", "autoTrimImages", and "autoTrimImages transformAttributes".

**Returns**
Nothing.

**Description**
Moves and resizes the selection in a single operation.

**dom.setSelectionMask()**

**Availability**
Fireworks 4.

**Usage**
dom.setSelectionMask(mask, howToCombineMasks)

**Arguments**
mask  Specifies the mask to be applied (see “Mask data type” on page 8). If mask is null, an existing pixel-selection mask is removed.

howToCombineMasks  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.

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

**dom setShowEdges()**

**Availability**
Fireworks 3.
Usage

\texttt{dom.setShowEdges(bShowEdges)}

Arguments

\texttt{bShowEdges} Boolean. If \texttt{true}, the Show Edges option is turned on; if \texttt{false}, the option is turned off.

Returns

Nothing.

Description

Specifies whether the Show Edges option is on or off.

dom.setShowGammaPreview()

Availability

Fireworks 3.

Usage

\texttt{dom.setShowGammaPreview(bPreviewGamma)}

Arguments

\texttt{bPreviewGamma} Boolean. If \texttt{true}, the Preview Gamma option is turned on; if \texttt{false}, the option is turned off.

Returns

Nothing.

Description

Specifies whether the Preview Gamma option is on or off.

dom.setShowGrid()

Availability

Fireworks 3.

Usage

\texttt{dom.setShowGrid(bShow)}

Arguments

\texttt{bShow} Boolean. If \texttt{true}, the grid is visible; if \texttt{false}, the grid is not visible.

Returns

Nothing.

Description

Specifies whether the grid is visible.
dom.setShowGuides()

Availability
Fireworks 3.

Usage
dom.setShowGuides(bShow)

Arguments
bShow Boolean. If true, the normal guides are visible; if false, the normal guides are not visible.

Returns
Nothing.

Description
Specifies whether normal guides are visible.

dom.setShowRulers()

Availability
Fireworks 3.

Usage
dom.setShowRulers(bShow)

Arguments
bShow Boolean. If true, the rulers are visible; if false, the rulers are not visible.

Returns
Nothing.

Description
Specifies whether rulers are visible.

dom.setShowSliceGuides()

Availability
Fireworks 3.

Usage
dom.setShowSliceGuides(bShow)

Arguments
bShow Boolean. If true, the slice guides are visible; if false, slice guides are not visible.
Returns
Nothing.

Description
Specifies whether slice guides are visible.

dom.setShowSliceOverlay()

Availability
Fireworks 3.

Usage
dom.setShowSliceOverlay(bShow)

Arguments
bShow Boolean. If true, the slice overlay is visible; if false, slice overlay is not visible.

Returns
Nothing.

Description
Specifies whether the slice overlay is visible.

dom.setSliceAutonaming()

Availability
Fireworks 3.

Usage
dom.setSliceAutonaming(bAutoname)

Arguments
bAutoname Boolean. If true, automatic naming is turned on for the slice; if false, automatic naming is turned off.

Returns
Nothing.

Description
If a single slice is selected, this function turns automatic naming on or off for the slice.

dom.setSliceType()

Availability
Fireworks 3; modified in Fireworks 10.
Usage
dom.setSliceType(slicetype)

Arguments
slicetype Specifies the slice type as image (foreground image), background image, or empty (HTML Slice).

Returns
Nothing.

Description
Sets the slice type as image (foreground image), background image, or empty (HTML Slice).

dom.setSliceExportOptions()

Availability
Fireworks 3.

Usage
dom.setSliceExportOptions(exportOptions)

Arguments
exportOptions An ExportOptions object (see “ExportOptions object” on page 260).

Returns
Nothing.

Description
Sets the export options for the selected slices.

dom.setSliceFilename()

Availability
Fireworks 3.

Usage
dom.setSliceFilename(fileURL)

Arguments
fileURL A string, which is expressed as a file://URL, that specifies the name to be given to the slice.

Returns
Nothing.

Description
If a single slice is selected, this function turns off automatic naming for the slice and sets its filename to the specified URL.
**dom.setSliceGuideColor()**

**Availability**
Fireworks 3.

**Usage**
dom.setSliceGuideColor(color)

**Arguments**
- **color** A color string (see "Color string data type" on page 8).

**Returns**
Nothing.

**Description**
Sets the color that is used to display slice guides. To set the color of normal guides, use `dom.setGuideColor()`.

**See also**
dom.setGuideColor()

dom.setSliceHtml()

**Availability**
Fireworks 3.

**Usage**
dom.setSliceHtml(htmlText)

**Arguments**
- **htmlText** A string that specifies the HTML text for the slice.

**Returns**
Nothing.

**Description**
If a single slice is selected, this function sets the slice’s HTML text.

dom.setSliceIsHtml()

**Availability**
Fireworks 3.

**Usage**
dom.setSliceIsHtml(bHtml)
Arguments
**bHtml** Boolean. If true, sets the slices as HTML; if false, sets the slices as Image.

Returns
Nothing.

Description
Sets the selected slices as HTML or Image.

**dom.setSnapToGrid()**

Availability
Fireworks 3.

Usage
```
dom.setSnapToGrid(bSnap)
```

Arguments
**bSnap** Boolean. If true, the tools snap to the grid; if false, they do not.

Returns
Nothing.

Description
Specifies whether tools snap to the grid.

**dom.setSnapToGuides()**

Availability
Fireworks 3.

Usage
```
dom.setSnapToGuides(bSnap)
```

Arguments
**bSnap** Boolean. If true, the tools snap to all guides; if false, they do not.

Returns
Nothing.

Description
Specifies whether tools snap to guides.
extending fireworks

The Document object

**dom.setSymbolProperties()**

*Availability*
Fireworks 3; modified in Fireworks 9.

*Usage*

dom.setSymbolProperties(currentName, symbolType, newName, status)

*Arguments*
- **currentName** A string value that 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).
- **symbolType** A string value. Acceptable values are "graphic", "button", and "animation".
- **newName** A string value that specifies the new name for the symbol.
- **status** A Boolean value that enables or disables 9-slice scaling on the specified symbol.

*Returns*
Nothing.

*Description*
Sets the name and symbol type of the specified symbol.

**dom.setTextAlignment()**

*Availability*
Fireworks 3.

*Usage*

dom.setTextAlignment(alignment)

*Arguments*
- **alignment** Acceptable values are "left", "center", "right", "justify", "stretch", "vertical left", "vertical center", "vertical right", "vertical justify", and "vertical stretch".

*Returns*
Nothing.

*Description*
Sets the alignment of the selected text items.

**dom.setTextAntiAliasing()**

*Availability*
Fireworks 3.
Usage

dom.setTextAntiAliasing(level)

Arguments

level Acceptable values are "crisp", "smooth", and "strong".

Returns

Nothing.

Description

Sets the anti-aliasing level for the selected blocks of text.

See also

dom.enableTextAntiAliasing()

**dom.setTextAutoKern()**

Availability

Fireworks 3.

Usage

dom.setTextAutoKern(bKern)

Arguments

bKern Boolean. If true, automatic kerning is on for the selected text items; if false, automatic kerning is off.

Returns

Nothing.

Description

Specifies whether automatic kerning is on or off for the selected text items.

**dom.setTextCharSpacing()**

Availability

Fireworks MX.

Usage

dom.setTextCharSpacing(charSpace)

Arguments

charSpace A floating-point percentage of the default space to add to (positive values) or remove from (negative values) two adjacent characters. To increase the spacing by 15%, for example, pass 0.15.

Returns

Nothing.
Description
Adjusts the kerning of text.

**dom.setTextCustomAntiAliasOverSample()**

Availability
Fireworks MX 2004.

Usage
```
dom.setTextCustomAntiAliasOverSample(overSample)
```

Arguments
- **overSample** The integer 4, 8 or 16 that specifies the amount of oversampling used to anti-alias text in custom mode.

Returns
Nothing.

Description
Sets the oversampling used to anti-alias text in custom mode.

**dom.setTextCustomAntiAliasSharpness()**

Availability
Fireworks MX 2004.

Usage
```
dom.setTextCustomAntiAliasSharpness(sharpness)
```

Arguments
- **sharpness** An integer from 0 to 255.

Returns
Nothing.

Description
Sets the sharpness value used to anti-alias text in custom mode.

**dom.setTextCustomAntiAliasStrength()**

Availability
Fireworks MX 2004.

Usage
```
dom.setTextCustomAntiAliasStrength(strength)
```

Last updated 5/17/2012
Arguments
Strength An integer value, from 0 to 255, for the amount of anti-aliasing to apply.

Returns
Nothing.

Description
Sets the strength value used to anti-alias text in custom mode.

dom.setTextFlow()

Availability
Fireworks 3.

Usage
dom.setTextFlow(flowDirection)

Arguments
flowDirection Acceptable values are "left to right" and "right to left".

Returns
Nothing.

Description
Sets the horizontal flow direction of the selected text items.

dom.setTextHorizontalScale()

Availability
Fireworks MX.

Usage
dom.setTextHorizontalScale(horizScale)

Arguments
horizScale A floating-point number that describes how much to scale the text characters horizontally. A value of 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.

Description
Sets the horizontal scaling of text. For vertical text mode, this function stretches or compresses the height of the characters.
**dom.setTextLeading()**

**Availability**
Fireworks MX.

**Usage**
```
dom.setTextLeading(leadingValue, leadingMode)
```

**Arguments**
- `leadingValue`: A floating-point number that determines the spacing between two lines of text. The meaning of `leadingValue` depends on `leadingMode`.
- `leadingMode`: Acceptable values are "exact" or "percentage". If set to "exact", `leadingValue` is the number of pixels between two lines of text. If set to "percentage", `leadingValue` is a percentage of the default leading; 1.0 is the default leading, 0.5 is half the default leading, and 2.0 is double the default leading.

**Returns**
Nothing.

**Description**
Sets the leading between lines of text. For vertical text mode, the leading is the space between two adjacent columns of text.

**dom.setTextOnPathMode()**

**Availability**
Fireworks 3.

**Usage**
```
dom.setTextOnPathMode(mode)
```

**Arguments**
- `mode`: Acceptable values are "rotate", "vertical", "skew vertical", and "skew horizontal".

**Returns**
Nothing.

**Description**
Determines how the selected text-on-a-path items are displayed.

**dom.setTextOnPathOffset()**

**Availability**
Fireworks 3.

**Usage**
```
dom.setTextOnPathOffset(offset)
```
Arguments
offset A floating-point value that specifies the offset distance, in pixels.

Returns
Nothing.

Description
Sets the offset value between the items in the selected text-on-a-path.

dom.setTextOrientation()

Availability
Fireworks 3.

Usage
dom.setTextOrientation(orientation)

Arguments
orientation Acceptable values are "horizontal left to right", "vertical right to left", "horizontal right to left", and "vertical left to right".

Returns
Nothing.

Description
Sets the horizontal/vertical text orientation of the selected text items.

dom.setTextParaIndent()

Availability
Fireworks MX.

Usage
dom.setTextParaIndent(paraIndent)

Arguments
paraIndent The number of pixels by which to indent the first line of a paragraph.

Returns
Nothing.

Description
Sets the paragraph indentation of text, in pixels.
dom.setTextParaSpacingAfter()

Availability
Fireworks MX.

Usage
dom.setTextParaSpacingAfter(paraSpaceAfter)

Arguments
paraSpaceAfter The number of pixels to place after a paragraph before starting the next paragraph.

Returns
Nothing.

Description
Sets the after-paragraph spacing for text; that is, the number of pixels to move down before starting the next paragraph. For vertical text mode, this function defines the vertical distance between paragraphs.

dom.setTextParaSpacingBefore()

Availability
Fireworks MX.

Usage
dom.setTextParaSpacingBefore(paraSpaceBefore)

Arguments
paraSpaceBefore The number of pixels to move down before starting a new paragraph.

Returns
Nothing.

Description
Sets the before-paragraph spacing for text; that is, the number of pixels to move down from the previous paragraph before starting the new paragraph. For vertical text mode, this function defines the vertical distance between paragraphs. If you apply dom.setTextParaSpacingAfter() in one paragraph, and dom.setTextParaSpacingBefore() in the second paragraph, the space between the two paragraphs would be the sum of both spacing arguments.

dom.setTextRuns()

Availability
Fireworks 3.

Usage
dom.setTextRuns(textRuns)
Arguments
textRuns A TextRuns object (see “TextRuns object” on page 275).

Returns
Nothing.

Description
Replaces the text in the selected text blocks with the styled text that is described by the TextRuns object passed in the argument.

**dom.setTransformMode()**

Availability
Fireworks 3.

Usage
dom.setTransformMode(mode)

Arguments
mode Acceptable values are "paths" and "pixels".

Returns
Nothing.

Description
Sets the transform mode for the selected text, instance items, or both.

**dom.setTextRectangle()**

Availability
Fireworks 3.

Usage
dom.setTextRectangle(boundingRectangle)

Arguments
boundingRectangle A rectangle that specifies the new size within which the text item should flow (see “Rectangle data type” on page 9).

Returns
Nothing.

Description
Changes the bounding rectangle of 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 in the new rectangle is not visible.
dom.setTextRectangleAuto()

Availability
Fireworks 3.

Usage
dom.setTextRectangleAuto()

Arguments
None.

Returns
Nothing.

Description
Recalculates the bounding rectangle of the selected text item, setting the rectangle to the smallest box that encloses the text.

See also
dom.setTextRectangleAutoFromPoint()

dom.setTextRectangleAutoFromPoint()

Availability
Fireworks 3.

Usage
dom.setTextRectangleAutoFromPoint(anchortPoint)

Arguments
anchortPoint  A point that specifies the x,y coordinates of the location at which the text box should be anchored (see “Point data type” on page 9). 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.

Description
Performs the same function as dom.setTextRectangleAuto(), but lets you pass a point to specify where the rectangle should be located.

See also
dom.setTextRectangleAuto()
**dom.setWebObjectsVisibility()**

*Availability*
Fireworks CS3.

*Usage*
dom.setWebObjectsVisibility(value)

*Arguments*

value A Boolean value that specifies the visibility of web objects. If the value is true, the web objects are visible. If the value is false the web objects are hidden.

*Returns*
Nothing.

*Description*
Sets the visibility of web objects on the current page.

**dom.shareLayerToPages()**

*Availability*
Fireworks CS3.

*Usage*
dom.shareLayerToPages(layerNum, addToPages, deleteFromPages)

*Arguments*

layerNum A long value that indicates the layer number for the layer that is to be shared across pages.

addToPages A comma-separated string value that specifies the names of all pages that are adding the specified layer.

deleteFromPages A comma-separated string value that specifies the names of all pages that are removing the specified layer.

*Returns*
Nothing.

*Description*
Shares or removes a specified foreground layer for specified pages.

**dom.showAllHidden()**

*Availability*
Fireworks 3.

*Usage*
dom.showAllHidden()
Arguments
None.

Returns
Nothing.

Description
Shows all the items that were hidden through `dom.hideSelection()`.

See also
`dom.hideSelection()`

**dom.snapToPixel()**

Availability
Fireworks CS5.

Usage
```
fw.getDocumentDOM().snapToPixel();
```

Arguments
None.

Returns
Nothing.

Description
This will run the `snapToPixel` command on the selected object(s).

**dom.splitPaths()**

Availability
Fireworks 3.

Usage
```
dom.splitPaths()
```

Arguments
None.

Returns
Nothing.

Description
Splits the selected paths. Compound paths are split into separate contours.
**dom.swapBrushAndFillColors()**

**Availability**
Fireworks 3.

**Usage**
dom.swapBrushAndFillColors()

**Arguments**
None.

**Returns**
Nothing.

**Description**
Swaps the current brush color and current fill color. This function has no effect on any selected items.

**dom.transformSelection()**

**Availability**
Fireworks 3, updated in Fireworks 4.

**Usage**
dom.transformSelection(matrix, options)

**Arguments**
matrix  A three-by-three transformation matrix (see “Matrix data type” on page 9).

options  Acceptable values, some of which were added in Fireworks 4, are ",", "transformAttributes",
"autoTrimImages","autoTrimImages transformAttributes", "rememberQuad", "transformAttributes
rememberQuad", "autoTrimImages rememberQuad", and "autoTrimImages transformAttributes
rememberQuad".

**Returns**
Nothing.

**Description**
Transforms the selection using the specified three-by-three matrix.

**dom.tween()**

**Availability**
Fireworks 3.

**Usage**
dom.tween()
Arguments

numSteps  An integer that specifies how many new instances are generated.

bDistribute  Boolean. If true, the new instances are distributed to states.

Returns

Nothing.

Description

Tweens between the two selected instances.

**dom.undo()**

Availability

Fireworks 3.

Usage

dom.undo()

Arguments

None.

Returns

Nothing.

Description

Undoes the most recent step performed, as long as that step is actually able to be undone; meaning, if you use a command that contains multiple JavaScript instructions, then you can undo the command (all 10 JavaScript instructions) and not just one JavaScript instruction within that command. Most (but not all) JavaScript functions cause an action to be executed that cannot be undone.

**dom.updateSymbol()**

Availability

Fireworks 3.

Usage

dom.updateSymbol

Arguments

name  The name of a 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.
Description
Updates the specified linked symbol.

dom.ungroup()

Availability
Fireworks 3.

Usage
dom.ungroup()

Arguments
None.

Returns
Nothing.

Description
Ungroups any grouped items in the selection. To group items, use dom.group().

See also
dom.group()

dom.unsetMasterPage()

Availability
Fireworks CS3.

Usage
dom.unsetMasterPage()

Arguments
None

Returns
Nothing.

Description
Resets the document’s master page. Makes the current master page into a normal page. For example:

fw.getDocumentDOM().unsetMasterPage()
Chapter 4: The Fireworks Object

The Fireworks object is a 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 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 8.*

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>activeTool</td>
<td>string</td>
<td>The active tool in the application.</td>
</tr>
<tr>
<td>activeViewState</td>
<td>float</td>
<td>The scaling (zoom value) of the active view. 1.0=100% of the normal view.</td>
</tr>
<tr>
<td>appBatchCodeDir•</td>
<td>string</td>
<td>The path to the Batch Code directory, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appDir•</td>
<td>string</td>
<td>The path to the directory that contains the Fireworks application, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appExportSettingsDir•</td>
<td>string</td>
<td>The path to the Export Settings directory, which is expressed as <code>file://URL</code>. In Fireworks, 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.</td>
</tr>
<tr>
<td>appFavoritesDir•</td>
<td>string</td>
<td>The path to the URL Libraries directory, which is expressed as <code>file://URL</code>. In Fireworks, 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.</td>
</tr>
<tr>
<td>appHelpDir•</td>
<td>string</td>
<td>The path to the directory that contains the Fireworks help file, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appHtmlCodeDir•</td>
<td>string</td>
<td>The path to the HTML Code directory, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appJsCommandsDir•</td>
<td>string</td>
<td>The path to the Commands directory, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appJsExtensionsDir•</td>
<td>string</td>
<td>The path to the JSExtensions directory, which is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>appMacCreator•</td>
<td>string</td>
<td>In the format: &quot;MKBY*&quot;</td>
</tr>
<tr>
<td>appMacJsFileFileType•</td>
<td>string</td>
<td>In the format: &quot;TEXT*&quot;</td>
</tr>
<tr>
<td>appName•</td>
<td>string</td>
<td>The name of the application. This attribute is part of the common API, so it also appears as <code>app.appName</code> (as implemented in Adobe Dreamweaver).</td>
</tr>
</tbody>
</table>
### The Fireworks Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>appPatternsDir</td>
<td>string</td>
<td>The path to the Patterns directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appPrefsDir</td>
<td>string</td>
<td>The path to the Preferences directory, which is expressed as a file://URL.</td>
</tr>
<tr>
<td>appPresetsDir</td>
<td>string</td>
<td>The path to the Presets directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In Fireworks, 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.</td>
</tr>
<tr>
<td>appSettingsDir</td>
<td>string</td>
<td>The path to the Settings directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appSmartShapesDir</td>
<td>string</td>
<td>The path to the application’s Auto Shapes directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appSmartShapeToolsDir</td>
<td>string</td>
<td>The path to the application’s Auto Shape Tools directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appStylesDir</td>
<td>string</td>
<td>The path to the Styles directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In Fireworks, 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.</td>
</tr>
<tr>
<td>appSwfCommandsDir</td>
<td>string</td>
<td>The path to the SWF Commands directory, which is expressed as a file://URL.</td>
</tr>
<tr>
<td>appSymbolLibrariesDir</td>
<td>string</td>
<td>The path to the Libraries directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appTemplatesDir</td>
<td>string</td>
<td>Returns the path of the Templates directory present at the location where you have installed Fireworks. Var tempLoc=fw.appTemplatesDir; alert(tempLoc);</td>
</tr>
<tr>
<td>appTexturesDir</td>
<td>string</td>
<td>The path to the Textures directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>appXtrasDir</td>
<td>string</td>
<td>The path to the Xtras directory, which is expressed as file://URL.</td>
</tr>
<tr>
<td>batchStatusString</td>
<td>string</td>
<td>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.</td>
</tr>
<tr>
<td>currentScriptDir</td>
<td>string</td>
<td>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).</td>
</tr>
<tr>
<td>currentScriptFileName</td>
<td>string</td>
<td>The filename of the currently running script (or could be null). This name is the script’s filename, not the full path.</td>
</tr>
<tr>
<td>dialogs</td>
<td>object</td>
<td>Provides access to an instance of the Dialogs class, which opens specific dialog boxes.</td>
</tr>
<tr>
<td>Property</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dismissBatchDialogWhenDone</td>
<td>Boolean</td>
<td>If set to true, Fireworks will automatically close the Batch Process dialog box when the script finishes. This function has no effect if the Batch Process dialog box does not appear.</td>
</tr>
<tr>
<td>documentList</td>
<td>array</td>
<td>Array of the current open Document objects (for more information, see &quot;The Document object&quot; on page 28). If no document is open, it returns an array of length zero.</td>
</tr>
<tr>
<td>documents</td>
<td>array</td>
<td>Array of the current open Document objects (for more information, see &quot;The Document object&quot; on page 28). If no document is open, returns an array of length zero.</td>
</tr>
<tr>
<td>ellipseBCPConst</td>
<td>float</td>
<td>A fixed value of 0.55229187012 used to calculate the distance between a point and its predecessor/successor for a perfect circle. For example, for a circle with a radius of 100 pixels, the predecessor/successor is 100*fw.ellipseBCPConst pixels away from the point itself.</td>
</tr>
<tr>
<td>errorReportingOK</td>
<td>Boolean</td>
<td>If set to true, Fireworks will allow posting an error while a script is running.</td>
</tr>
<tr>
<td>files</td>
<td>object</td>
<td>The FilesClass object used to perform file operations (open, close, delete, and so on).</td>
</tr>
<tr>
<td>getDynamicSWFURL</td>
<td>string</td>
<td>Returns the location of the SWF file.</td>
</tr>
<tr>
<td>getEndBackgroundColor</td>
<td>color</td>
<td>Returns the end color for the background gradient. This function is only useful for the Windows platform.</td>
</tr>
<tr>
<td>getStartBackgroundColor</td>
<td>color</td>
<td>Returns the start color for the background gradient. This function is only useful for the Windows platform.</td>
</tr>
<tr>
<td>historyPalette</td>
<td>object</td>
<td>History panel object. There are no DOM properties for the History panel, only API calls. For more information, see &quot;History panel functions&quot; on page 324.</td>
</tr>
<tr>
<td>isConnectedToInternet</td>
<td>integer</td>
<td>Returns whether the operating system is connected to the Internet. The Start Page has a dynamic content panel that loads content from the Internet. The Start Page queries this property before attempting to download the dynamic content.</td>
</tr>
<tr>
<td>mruRecentFilesList</td>
<td>array</td>
<td>Array of recent open files. If there are no open files, returns an array length of zero.</td>
</tr>
<tr>
<td>mruRecentFileNames</td>
<td>array</td>
<td>Array of recent open file names. If there are no open files, returns an array length of zero.</td>
</tr>
<tr>
<td>platform</td>
<td>string</td>
<td>The string &quot;mac&quot; if Fireworks is running on the Macintosh, or &quot;win&quot; if running on Windows.</td>
</tr>
<tr>
<td>progressCountCurrent</td>
<td>integer</td>
<td>The first number (x) that appears in the Batch Progress dialog box, in the &quot;File x of y&quot; field. Set this property to change the number.</td>
</tr>
<tr>
<td>progressCountTotal</td>
<td>integer</td>
<td>The second number (y) that appears in the Batch Progress dialog box, in the &quot;File x of y&quot; field. Set this property to change the number.</td>
</tr>
<tr>
<td>screenRect</td>
<td>rectangle</td>
<td>The size of the main screen on this computer, in pixels. Useful for positioning windows or panels.</td>
</tr>
</tbody>
</table>
The Fireworks Object

In Fireworks CS5, `fw` is synonymous with the Fireworks object. All methods of the Fireworks object can be referred to as `fireworks.functionName()` or as `fw.functionName()`.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>selection</td>
<td>array</td>
<td>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 <code>null</code>.</td>
</tr>
<tr>
<td>selectedMask</td>
<td>object</td>
<td>If a single item is selected and that item is a mask, this property returns an ElementMask object (for more information, see “ElementMask object” on page 259); otherwise, it returns <code>null</code>.</td>
</tr>
<tr>
<td>styles</td>
<td>array</td>
<td>Array of the Style object that is currently loaded in the Style panel (for more information, see “Style object” on page 273).</td>
</tr>
<tr>
<td>textInsertionIndex</td>
<td>integer</td>
<td>Insertion index into the current active text object. If there is no text selected, returns a value of -1.</td>
</tr>
<tr>
<td>textInsertionLength</td>
<td>integer</td>
<td>Insertion length into the current active text object. If there is no text selected, returns a value of -1.</td>
</tr>
<tr>
<td>textOutputEncoding</td>
<td>string</td>
<td>The default text encoding for any text file that the JavaScript interpreter generates. Use &quot;iso-8859-1&quot; for ASCII or &quot;utf-8&quot; for Unicode.</td>
</tr>
<tr>
<td>userJsCommandsDir</td>
<td>string</td>
<td>The path to the user-level Commands directory, which is expressed as a file://URL. In Fireworks, 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.</td>
</tr>
<tr>
<td>userSmartShapesDir</td>
<td>string</td>
<td>The path to the user's Auto Shapes directory, which is expressed as a file://URL.</td>
</tr>
<tr>
<td>userShapeToolsDir</td>
<td>string</td>
<td>The path to the user’s Auto Shape Tools directory, which is expressed as a file://URL.</td>
</tr>
<tr>
<td>userSymbolLibrariesDir</td>
<td>string</td>
<td>The path to the user's Symbol Libraries, which is expressed as file:///URL.</td>
</tr>
<tr>
<td>userSwfCommandsDir</td>
<td>string</td>
<td>The path to the user-level SWF Commands directory, which is expressed as a file://URL. In Fireworks, 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.</td>
</tr>
<tr>
<td>xhtmlFormat</td>
<td>Boolean</td>
<td>Determines whether the JavaScript interpreter should output XHTML formatted files or HTML formatted files; XHTML(true) or HTML(false).</td>
</tr>
</tbody>
</table>
fw.browseDocument()

Availability
Fireworks 3.

Usage
fw.browseDocument(URL)

Arguments
URL  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.

Description
Opens the user’s primary browser and displays the specified URL.

fw.browseForFileURL()

Availability
Fireworks 3.

Usage
fw.browseForFileURL(browseType, title, previewArea)

Arguments
browseType  Acceptable values are "open", "select", and "save". The first two values display an Open dialog box; each is acceptable for compatibility with Adobe Dreamweaver. The third value displays a Save dialog box.

title and previewArea  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.

Description
Displays an Open or Save dialog box to the user.

fw.browseForFolderURL()

Availability
Fireworks 3.

Usage
fw.browseForFolderURL({title}, {startFolder})
Arguments

**title**  An optional string that specifies a title for the dialog box that appears. If it is omitted or `null`, a default title appears.

**startFolder**  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.

Description
Displays a dialog box that lets a user select a particular directory.

**fw.browseHelp()**

Availability
Fireworks MX.

Usage

```javascript
fw.browseHelp(helpID)
```

Arguments

**helpID**  The index number of the help topic to view.

Returns
Nothing.

Description
Opens the specified help topic in the help viewer.

**fw.checkFwJsVersion()**

Availability
Fireworks 3.

Usage

```javascript
fw.checkFwJsVersion(version)
```

Arguments

**version**  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.

Description
Checks the JavaScript API for incompatibilities.
**fw.chooseBrowser()**

Availability
Fireworks MX.

Usage
fw.chooseBrowser(primaryBrowser)

Arguments
primaryBrowser  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.

Description
Displays a dialog box that lets the user select a primary or secondary browser.

**fw.chooseScriptTargetDialog()**

Availability
Fireworks 4.

Usage
fw.chooseScriptTargetDialog(formatlist)

Arguments
formatlist  A list of target documents for an operation. Its use is similar to that in fw.locateDocDialog(), except that formatlist is required, and you cannot specify a maximum number of documents.

Returns
An array of file://URLs, or null if the dialog box is canceled.

Description
Displays a dialog box that lets the user choose the target documents for an operation. The dialog box lets the user specify currently open files, files in the project list, or files that are explicitly selected.

See also
fw.locateDocDialog()

**fw.closeDocument()**

Availability
Fireworks 3.
Usage
fw.closeDocument(document, {bPromptToSaveChanges})

Arguments
document A Document object that specifies the document to close (see “The Document object” on page 28).
bPromptToSaveChanges An optional Boolean argument. 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.

Description
Closes the specified document.

fw.createDocument()

Availability
Fireworks 3.

Usage
fw.createDocument().

Arguments
None.

Returns
The Document object for the newly created document (see “The Document object” on page 28).

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().

See also
fw.createFireworksDocument()

fw.createDocumentWithDialog()

Availability
Fireworks MX 2004.

Usage
fw.createDocumentWithDialog()

Arguments
None.
Returns
The Document object for the newly created document (see “The Document object” on page 28).

Description
Shows the New Document dialog box and allows the user to create a new document.

fw.createFireworksDocument()

Availability
Fireworks 3.

Usage
fw.createFireworksDocument(size, res, backgroundColor)

Arguments
size A point whose x value specifies the document’s width and whose y value specifies the document’s height. Both values are in pixels.
res Specifies the resolution for the scaled document (see “Resolution data type” on page 9).
backgroundColor A color string (see “Color string data type” on page 8).

Returns
The Document object for the newly created document (see “The Document object” on page 28).

Description
Opens a new document and selects it. Values for size, resolution, and color are explicitly specified. To open a new document with the current default values, use fw.createDocument().

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");

See also
fw.createDocument()

fw.currentWorkspaceName()

Availability
Fireworks 10.

Usage
fw.currentWorkspaceName()

Arguments
None.
Returns
The name of the current workspace layout.

Description
Returns the name of the current workspace layout.

fw.reloadCommonLibrary()

Availability
Fireworks 9.

Usage
fw.reloadCommonLibrary()

Arguments
None.

Returns
Reloads the Common Library if you add a new widget, etc. You can reload the library from the JSF as well.

fw.dialogs.runEditGrids()

Availability
Fireworks 3.

Usage
fw.dialogs.runEditGrids()

Arguments
None.

Returns
Opens the Edit Grids dialog box.

fw.dialogs.runEditGuides()

Availability
Fireworks 3.

Usage
fw.dialogs.runEditGuides()

Arguments
None.
Returns
Opens the Edit Guides dialog box.

fw.dialogs.runNumericTransform()

Availability
Fireworks 3.

Usage
fw.dialogs.runNumericTransform()

Arguments
None.

Returns
Opens the Numeric Transform dialog box.

fw.disableFlashDebugging()

Availability
Fireworks MX

Usage
fw.disableFlashDebugging()

Arguments
None.

Returns
Nothing.

Description
Turns off debugging messages for Flash commands. For a description of the Flash debugging capabilities, see “fw.enableFlashDebugging()” on page 202. For more information about constructing Flash command panels for Fireworks, see “Flash panels” on page 299.

fw.dismissBatchDialogWhenDone()
Arguments

**autoClose**  A Boolean value. If set to **true**, the Batch Progress dialog box closes automatically (without user intervention) when the script finishes.

Returns

Nothing.

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.

**fw.enableFlashDebugging()**

Availability

Fireworks MX

Usage

`fw.enableFlashDebugging()`

Arguments

None.

Returns

Nothing.

Description

Turns on debugging messages for Flash commands. When Flash debugging is enabled, Fireworks displays the command string in a dialog box every time a Flash command calls `MMExecute()`. The `fw.enableFlashDebugging()` function is particularly useful for monitoring which commands are executed in a command panel. For information on how to turn off Flash debugging, see `fw.disableFlashDebugging()`. For more information about constructing Flash command panels for Fireworks, see “Flash panels” on page 299.

*Note: This debugging command works even if you are running a JavaScript file.*

**fw.exportAndCopyHTMLCode()**

Availability

Fireworks MX.

Usage

`fw.exportAndCopyHTMLCode(document)`

Arguments

**document**  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.

Description
Displays the export dialog box, which is preconfigured to export HTML and images and to copy the HTML code to the clipboard.

**fw.exportCSSLayers()**

Availability
Fireworks 4.

Usage
fw.exportCSSLayers(doc, mode, htmlURL, imagesURL, trim)

Arguments
docObject A Document object that specifies the document that contains the layers to export (see “The Document object” on page 28). To export layers from the current document, pass null.
htmlURL HTML folder path.
mode Can take values 0-3 denoting Multiple Layers, Multiple Slices, Multiple Frames, and Multiple Images respectively.
imagesURL Path of the images folder.
trim True if the document trims; false otherwise.

Returns
A Boolean value: true if successful; false otherwise.

Description
Exports document as CSS layers and images. The image names are based on the names in the Layers panel.

**fw.exportDirectorAsLayers()**

Availability
Fireworks MX.

Usage
fw.exportDirectorAsLayers(document, fileURL)

Arguments
document 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.

Description
Exports the specified document to the specified file as layers to be imported into Adobe Director.

fw.exportDirectorAsSlices()

Availability
Fireworks MX.

Usage
fw.exportDirectorAsSlices(document, fileURL)

Arguments
document 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.

Description
Exports the specified document to the specified file as Adobe Director images.

fw.exportDocumentAs()

Availability
Fireworks 3.

Usage
fw.exportDocumentAs(document, fileURL, exportOptions)

Arguments
document 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 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 An ExportOptions object (see “ExportOptions object” on page 260). 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
A Boolean value: true if successful; false otherwise.
Description
Exports the specified document to the specified file.

See also
fw.exportHtmlAndImages()

fw.exportFrames()

Availability
Fireworks 4.

Usage
fw.exportFrames(docObject, directoryURL)

Arguments
docObject A Document object that specifies the document that contains the frames to export (see “The Document object” on page 28). To export frames from the current document, pass null.
directoryURL The directory where the images will be placed, which is expressed as a file://URL.

Returns
A Boolean value: true if successful; false otherwise.

Description
Exports a document’s frames as individual images. The image names are based on the names in the Frames panel.

Example
The following command exports the frames in the current document to the C:\images directory:
fw.exportFrames(null, "file:///C|/images");

fw.exportFXG()

Availability
Fireworks CS5.

Usage
fw.exportFXG(doc, pages, fileUrl);

Arguments
doc specifies the document to be exported. The document that is currently open in Fireworks is exported if null is specified for this parameter.
pages “all pages” or “current page” or “selected objects”.
fileUrl “Specifies the file location to which it has to be exported.

Returns
A Boolean value: true if successful; false otherwise.
Description
Exports current page, all pages of the open document in Fireworks, or selected objects from the current page as FXG files.

fw.exportHtmlAndImages()

Availability
Fireworks 4.

Usage
fw.exportHtmlAndImages(doc, htmlUrl, imagesUrl)

Arguments
doc A Document object that specifies the document to be exported (see “The Document object” on page 28). If doc is null, the active document is exported.
htmlUrl The filename of the exported HTML file, which is expressed as a file://URL. If htmlUrl is null, no HTML is generated.
imagesUrl The name of the file containing 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 name of the image file. If multiple sliced images are exported, it uses imagesURL to generate automatically named images, and all images are placed in this directory.

Returns
A Boolean value: true if successful; false otherwise.

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.

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");

See also
fw.exportDocumentAs()

fw.exportIllustrator()

Availability
Fireworks MX.

Usage
fw.exportIllustrator(document, fileURL)
The Fireworks Object

Arguments

document: 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.

Description

Exports the specified document to the specified file in Adobe Illustrator format.

`fw.exportLayers()`

Availability

Fireworks 4.

Usage

`fw.exportLayers(docObject, directoryURL)`

Arguments

docObject: A Document object that specifies the document that contains the layers to export (see “The Document object” on page 28). To export layers from the current document, pass `null`.

directoryURL: The directory in which the images will be placed, which is expressed as a file://URL.

Returns

A Boolean value: `true` if successful; `false` otherwise.

Description

Exports a document’s layers as individual images. The image names are based on the names in the Layers panel. The layers from the current frame are exported.

Example

The following command exports the layers in the third open document to the C:\images directory.

```javascript
fw.exportLayers(fw.documents[2], "file:///C|/images");
```

`fw.exportMXMLAndImages()`

Availability

Fireworks 9.

Usage

`fw.exportMXMLAndImages(doc, htmlUrl, imagesUrl)`
Arguments
doc  A Document object that specifies the document to be exported (see “The Document object” on page 28). If doc is null, the active document is exported.

htmlUrl  The filename of the exported MXML file, which is expressed as a file://URL. If htmlUrl is null, no MXML is generated.

imagesUrl  The name of the file containing 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 name of the image file. If multiple sliced images are exported, it uses imagesURL to generate automatically named images, and all images are placed in this directory.

Returns
A Boolean value: true if successful; false otherwise.

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 MXML. The document is exported using the current export settings and export options.

Example
The following command exports the current document as HTML and as one or more images.

```
fw.exportMXMLAndImages(null, "file:///C|/mysite/nav.htm",
"file:///C|/mysite/images/nav.gif");
```

See also
fw.exportDocumentAs()

fw.exportPages()

Availability
Fireworks 11.

Usage
fw.exportPages(doc object,"Images/FPNG","Current/All",fileURL)

Arguments
Images  settings from the Optimize panel are used to export the page(s).

Values  Default argument values are “Images”, and “Current”.

Returns
None.

Description
Exports the specified pages in the specified format.

Example
fw.exportPages(null,"FWPNG","All","file:///Snow Leopard/a/test");
It exports all pages in the current document in Fireworks PNG format under the location /Snow Leopard/ folder name. The "test" at the end is used to signify a file. Page names are used to assign filenames.

**fw.exportPDF()**

**Availability**
Fireworks 10.

**Usage**
`fw.exportPDF(doc, pdfURL)`

**Arguments**
- `doc` Document object (for example, `fw.documents[2]`) that specifies the document to be exported. To export frames from the current document, pass null.
- `pdfURL` Specifies the filename for the exported file. If `pdfURL` is null, Fireworks displays the Export dialog box.

**Returns**
A Boolean value: true if successful; false otherwise.

**Description**
Exports the specified document to the specified file in PDF format.

**fw.exportPSD()**

**Availability**
Fireworks 4.

**Usage**
`fw.exportPSD(docObject, PSDDocumentURL)`

**Arguments**
- `docObject` A Document object that specifies the document to export (see “The Document object” on page 28). To export the current document, pass null.
- `PSDDocumentURL` The name of the Photoshop document to be created, which is expressed as a file://URL.

**Returns**
A Boolean value: true if successful; false otherwise.

**Description**
Exports a Fireworks document as a Photoshop document.

**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.

Usage
fw.exportSWF(docObject, FlashDocumentURL)

Arguments
docObject A Document object that specifies the document to be exported (see “The Document object” on page 28).
To export the current document, pass null.

FlashDocumentURL The name of the Adobe Flash document to be created, which is expressed as a file://URL.

Returns
A Boolean value: true if successful; false otherwise.

Description
Exports a Fireworks document as a Adobe Flash document.

Example
The Adobe 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.
EXTENDING FIREWORKS

The Fireworks Object

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 if SwfExportAllFrames 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.

Usage
fw.findApp(macAppSignature or winExeRegistryName)

Arguments
macAppSignature A Macintosh-specific string that identifies the signature of the application to find, such as "MKBY".

winExeRegistryName 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(). If no such application can be found, the URL is empty.
Description
Attempts to find the path to the requested application. On the Macintosh, Fireworks looks for the application using a four-character signature code. In Windows, Fireworks looks in the Windows registry under \HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths.

See also
fw.launchApp()

fw.findNext()

Availability
Fireworks 3.

Usage
fw.findNext()

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.

Description
Finds the next instance of the current search string and selects that section of the document. To begin a search, use fw.setUpFindReplace().

See also
fw.setUpFindReplace()

fw.findOpenDocument()

Availability
Fireworks 3.

Usage
fw.findOpenDocument(docname)

Arguments
docname A string that specifies the name of the document, which is expressed as file:///URL.

Returns
If the document is open, returns the Document object; otherwise, returns null (see “The Document object” on page 28).
Description
Determines whether the specified file is open in a Fireworks Document window.

fw.getBlue()

Availability
Fireworks MX.

Usage
fw.getBlue(color)

Arguments

color  RGB value of the color.

Returns
The hex value of blue color in the selected object.

Description
Returns the hex value of blue color in the selected object.

fw.getDocumentDOM()

Availability
Fireworks 3.

Usage
fw.getDocumentDOM({which-string})

Arguments

which-string  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.

Description
Gets the Document object for the active document (see “The Document object” on page 28).

fw.getDocumentDOM().combineCompoundShape

Availability
Fireworks CS5.

Usage
fw.getDocumentDOM().combineCompoundShape();
Other supported values: punch, union, crop, none.

**Arguments**
None.

**Returns**
None.

**Description**
Combines the compound shape that is selected. Requires selection of a compound shape.

**fw.getDocumentDOM().createCompoundShape**

**Availability**
Fireworks CS5.

**Usage**
`dom.createCompoundShape(operation)`

Other supported values: punch, union, crop, and none.

**Arguments**
Vector operation applied on paths created after applying this API.

**Returns**
None.

**Description**
Creates a compound shape from the selected vectors applying the operation that is specified. Requires two or more vectors to be selected.

**fw.getDocumentDOM().InsertPageForImport**

**Availability**
Fireworks CS5.

**Usage**
`fw.getDocumentDOM().InsertPageForImport(filepath, pageNum)`

**Arguments**
filepath URL of the PNG file.

pageNum The page number from which the new empty page is derived. The new page takes the name and bounds of this page in the source file.

**Returns**
None.
Description
A new page with name and bounds is created for the corresponding page number specified.

fw.getDocumentDOM().VectorOperation

Availability
Fireworks CS5.

Usage
fw.getDocumentDOM().VectorOperation = "intersect";

Other supported values: punch, union, crop, none.

Arguments
None.

Returns
None.

Description
Vector operation is applied on paths created after applying this API. A compound shape is created.

fw.getDocumentPath()

Availability
Fireworks 3.

Usage
fw.getDocumentPath(document)

Arguments
document 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.

Description
Gets the path and filename of the specified document.

fw.getFamilyNameForPSFont()

Availability
Fireworks 10.
Usage
fw.getFontFamilyNameForPSFont(font)

Arguments
font  Name of the PostScript font.

Returns
The family name of the specified PostScript font.

Description
Gets the family name of a PostScript font.

fw.getFloaterGroupings()

Availability
Fireworks 3.

Usage
fw.getFloatGroupings()

Arguments
None.

Returns
An array like the one in the following example:

[["stroke", "fill", "effect"], ["layers", "frames", "object"], ["mixer", "options", "swatches", "info"], ["styles", "library"], ["find", "project log"], ["url"], ["optimize", "optimized colors"], ["behaviors"], ["history"]]

Note: Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are "outside" the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be "mypanel".

Description
Gets an array of arrays that indicates the tab-grouping of the panels (even hidden ones).

fw.getFloaterPosition()

Availability
Fireworks 3.

Usage
fw.getFloaterPosition(panelName)
EXTENDING FIREWORKS

The Fireworks Object

Arguments

panelName Acceptable values are "find", "project log", "object", "info", "url", "effect", "history", "mixer", "fill", "stroke", "swatches", "layers", "frames", "behaviors", "optimize", "library", "styles", "optimized colors", "options", and "toolbox".

Note: Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are "outside" the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be “mypanel”.

Returns

A rectangle that specifies the bounds of the panel (see “Rectangle data type” on page 9).

Description

Gets the screen position and size of the specified panel.

fw.getFloaterVisibility()

Availability

Fireworks 3.

Usage

fw.getFloaterVisibility(panelName)

Arguments

panelName Acceptable values are "find", "project log", "object", "info", "url", "effect", "history", "mixer", "fill", "stroke", "swatches", "layers", "frames", "behaviors", "optimize", "library", "styles", "optimized colors", "options", and "toolbox".

Note: Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are "outside" the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be “mypanel”.

Returns

A Boolean value: true if the specified panel is visible, false otherwise.

Description

Determines whether a specified panel is visible.

fw.getGreen()

Availability

Fireworks MX.

Usage

fw.getGreen(color)
Arguments

color  The RGB value of the color.

Returns

Hex value of green color in the selected object.

Description

Returns the hex value of green color in the selected object.

fw.getHideAllFloaters()

Availability

Fireworks 3.

Usage

fw.getHideAllFloaters()

Arguments

None.

Returns

A Boolean value: true if the panels are hidden; false otherwise.

Description

Returns the hidden or visible status of the panels.

fw.getHTMLFileForScript()

Availability

Fireworks MX.

Usage

fw.getHTMLFileForScript()

Arguments

None.

Returns

A file URL.

Description

Returns an HTML file.
fw.getNumberOfTables()

Availability
Fireworks MX.

Usage
fw.getNumberOfTables(filename)

Arguments
filename 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.

Description
Returns the number of top-level (that is, non-nested) tables in a document.

fw.getPlatformNameForPSFont()

Availability
Fireworks CS5.

Usage
fw.getPlatformNameForPSFont(psFontName)

Arguments
psFontName The name of the PostScript font.

Returns
Platform name of the PostScript font.

Description
Returns the platform name of the PostScript font.

fw.getPref()

Availability
Fireworks 3.

Usage
fw.getPref(prefkey)

Arguments
prefkey 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.

Description
Returns the Preference value (string or numeric) that is associated with the specified Preference key.

See also
fw.setPref()

fw.getRed()

Availability
Fireworks MX.

Usage
fw.getRed(color)

Arguments
color  RGB value of the color.

Returns
The hex value of red color in the selected object.

Description
Returns the hex value of red color in the selected object.

fw.getStyleNameForPSFont()

Availability
Fireworks CS5.

Usage
fw.getStyleNameForPSFont(font)

Arguments
font  Name of the PostScript font.

Returns
The style name of the specified PostScript font.

Description
Returns the style name of the PostScript font.
fw.internalNameToUIName()

Availability
Fireworks MX2.

Usage
fw.internalNameToUIName(internalname)

Arguments
internalname  Internal name to be mapped to a UI name.

Returns
UI name string.

Description
Converts internal names (such as fn_Normal) which do no change across languages to UI names (such as Normal) which can change across languages. The UI name is what gets displayed to the user. Returns the internal name if no conversion is found.

fw.launchApp()

Availability
Fireworks MX.

Usage
fw.launchApp(appPath, filePathsToOpen)

Arguments
appPath    A file URL that specifies the executable to start. Typically, this value can be obtained by calling fw.findApp().

filePathsToOpen  An array of file URLs to open in the executable to start. It is safe to pass an empty array.

 Returns
A Boolean value that indicates whether the application started successfully.

Description
Starts an application using a file URL that is returned by fw.findApp(). You can specify, optionally, files to open in the application.

See also
fw.findApp()
fw.launchBrowserTo()

Availability
Fireworks MX.

Usage
fw.launchBrowserTo(url)

Arguments
url The URL to open in the primary web browser.

Returns
Nothing.

Example
The following command starts a browser that opens to the Adobe website:

fw.launchBrowserTo("http://www.Adobe.com");

Description
Starts Fireworks' primary web browser to open a URL.

Note: Use fw.launchBrowserTo() instead of getURL(). getURL() doesn't work as intended on Mac if Fireworks 3.6 is the default browser.

fw.locateDocDialog()

Availability
Fireworks 4.

Usage
fw.locateDocDialog(maxnumdocs, formatlist)

Arguments
maxnumdocs Specifies the maximum number of documents to choose.

formatlist A list of acceptable file types to open. The formatlist argument 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ADOBE AI3&quot;</td>
<td>Adobe Illustrator</td>
</tr>
<tr>
<td>&quot;Fireworks JavaScript&quot;</td>
<td>Fireworks JSF</td>
</tr>
<tr>
<td>&quot;kMoaCfFormat_BMP&quot;</td>
<td>Bitmap</td>
</tr>
<tr>
<td>&quot;kMoaCfFormat_FreeHand7and8&quot;</td>
<td>Adobe FreeHand 7 or 8</td>
</tr>
</tbody>
</table>
Returns
An array of file:// URLs, or null if the dialog box is canceled.

Description
Displays a dialog box that lets the user select one or more files.

fw.makeRGBColor()

Availability
Fireworks MX.

Usage
fw.makeRGBColor(red, green, blue)

Arguments
red Decimal value of red color.
green Decimal value of green color.
blue Decimal value of blue color.

Returns
RGB pixel value.

Description
Combines the individual hex values into a single RGB value.

fw.moveFocusToDoc()

Availability
Fireworks 8.

Usage
fw.moveFocusToDoc()
EXTENDING FIREWORKS
The Fireworks Object

Arguments
None.

Returns
Nothing.

Description
Moves the keyboard focus to the document window.

fw.newDocumentFromTemplate(arg1)

Availability
Fireworks CS5.

Usage
fw.newDocumentFromTemplate("file:///C:/file1.png");

Arguments
Arg1  Absolute path of the file which is to be used as a template.

Returns
None.

Description
Opens up a file for selecting any template from the pre-defined Templates folder in Fireworks installation directory. Specifying null in the argument will open the Template open dialog.

See also
fw.saveDocumentAsTemplate(arg1,arg2)

fw.newMobileDocument(arg1,arg2,arg3,arg4)

Availability
Fireworks CS5.

Usage
fw.newMobileDocument("ssss",100,100,100);

Arguments
Arg1  Device Name (Ignored by Fireworks)
Arg2  Width.
Arg3  Height.
Arg4  Resolution.
RETURNS
None.

DESCRIPTION
Creates a mobile document with the width, height, and resolution details.

SEE ALSO
fw.saveDocumentAsTemplate(arg1, arg2)

fw.openDocument()

AVAILABILITY
Fireworks 3, updated in Fireworks 4 and Fireworks 8.

USAGE
fw.openDocument({fileURL}, {bOpenAsNew}, {bOpenWithWindowHidden})

ARGUMENTS
fileURL  A string or an array of strings, each expressed as a file://URL, that specifies the files to be opened. This argument is optional. If fileURL is omitted or null, the Open Document dialog box appears.

bOpenAsNew  If bOpenAsNew, which was added in Fireworks 4, is true, the files are opened as unsaved and untitled documents. If bOpenAsNew is false (the default value), they are opened with their original names. This argument is optional.

bOpenWithWindowHidden  Boolean. If bOpenWithWindowHidden, which was added in Fireworks 8, is true, and if there is only one document to open, the document will be hidden when opened. If more than one document is being opened, this parameter is ignored. The default value is false. This parameter was added to enhance the Batch Process feature.

RETURNS
If any of the files can be opened, returns the Document object for each file. Returns null if none of the documents can be opened.

DESCRIPTION
Opens the specified files in new document windows. If a file is already open, it opens again; to avoid redundant open operations, call findOpenDocument() first.

SEE ALSO
fw.findOpenDocument()

fw.openMultiDocument()

AVAILABILITY
Fireworks 6.

USAGE
fw.openMultiDocument()
EXTENDING FIREWORKS
The Fireworks Object

Arguments
None.

Returns
Nothing.

Description
Brings up the Open dialog. Users can select and open multiple files in new document windows. If a file is already open, it opens again; to avoid redundant opens, call findOpenDocument() first.

See also
fw.findOpenDocument()

fw.popupColorPicker()

Availability
Fireworks MX.

Usage
fw.popupColorPicker(screenLoc, initialColor, allowTransparent, forceWeb216)

Arguments
screenLoc  The location at which the dialog box appears, in the form of a point {x:float, y:float} (for syntax details, see “Point data type” on page 9).
initialColor  The initially selected color in the dialog box, in the form #rrggbbaa (for syntax details, see “Color string data type” on page 8).
allowTransparent  A Boolean value that lets the user select a transparent color; set to true for transparent, false otherwise.
forceWeb216  A Boolean value that forces the specified color to fall within the web216 panel; set to true to force the color change, false otherwise.

Returns
The specified color in #rrggbbaa format (for syntax details, see “Color string data type” on page 8).

Description
Opens the pop-up color swatches dialog box to let the user select a color.

fw.popupColorPickerOverMouse()

Availability
Fireworks MX.

Usage
fw.popupColorPickerOverMouse(initialColor, allowTransparent, forceWeb216)
**Arguments**

- **initialColor** A color string in #rrggbbbaa format, which is the color initially selected in the dialog box. (For syntax details, see “Color string data type” on page 8.)

- **allowTransparent** A Boolean value that lets the user select a transparent color; set to true for transparent, false otherwise.

- **forceWeb216** 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 specified color in #rrggbbbaa format (For syntax details, see “Color string data type” on page 8).

**Description**
Opens the color pop-up window at the current mouse location to let the user select a color.

### fw.previewInDeviceCentral()

**Availability**
Fireworks CS5

**Usage**
fw.previewInDeviceCentral();

**Arguments**
None

**Returns**
None

**Description**
Launches Adobe Device Central and previews the current document in it. If there is no active document, an alert message is generated.

### fw.quit()

**Availability**
Fireworks 4.

**Usage**
fw.quit()

**Arguments**
None.

**Returns**
Nothing.
Description
Quits Fireworks, but prompts the user to save any changed documents before exiting. Identical to `fw.quitApplication()`.

fw.quitApplication()

Availability
Fireworks 3.

Usage
`fw.quitApplication()`

Arguments
None.

Returns
Nothing.

Description
Quits Fireworks, but prompts the user to save any changed documents before exiting.

fw.readNthTable()

Availability
Fireworks MX.

Usage
`fw.readNthTable(filename, tablenumber)`

Arguments
filename A fileURL for the file that contains the desired table.
tablenumber A long integer that specifies the desired table; the tables are zero-indexed.

Returns
A database that is constructed from the table data.

Description
Reads the specified table. The tables are zero-indexed.

fw.readPanelStateFromFile()
Usage
fw.readPanelStateFromFile(filepath)

Arguments
filepath The location of the panel state file as a string in the format file://URL.

Returns
Nothing.

Description
Reads in a panel state file, which is generated by `fw.writePanelStateToFile()`, and moves the panels, Property inspector, and toolbox to the appropriate locations.

**fw.reloadCommonLibrary()**

Availability
Fireworks 9.

Usage
fw.reloadCommonLibrary()

Arguments
None.

Returns
Nothing.

Description
Reloads the Common Library. If the user adds new widget, the library can be reloaded from the JSF as well.

**fw.replace()**

Availability
Fireworks 3.

Usage
fw.replace()

Arguments
None.

Returns
The number of items that are replaced, or –1 if there are items in the document that remain to be searched.

Description
Verifies that the selection matches the current search string and replaces it with the replacement string.
See also
fw setUpFindReplace()

fw.replaceAll()

Availability
Fireworks 3.

Usage
fw.replaceAll()

Arguments
None.

Returns
The number of items replaced, or –1 if the search is not yet complete.

Description
Performs a replace all operation on the active document using the current search-and-replace strings.

See also
fw setUpFindReplace()

fw.resetWarnings()

Availability
Fireworks 4.

Usage
fw.resetWarnings()

Arguments
None.

Returns
Nothing.

Description
Resets all warnings that have a "Don’t show me again" checkbox to popup when encountered by the user.

fw.revertDocument()

Availability
Fireworks 3.
Usage
fw.revertDocument({document})

Arguments
document  A Document object, for example, fw.documents[2], that specifies the document to be reverted. This argument is optional. If document is omitted or null, the active document is reverted.

Returns
Nothing.

Description
Restores the specified document to its previously saved version.

fw.runScript()

Availability
Fireworks 3.

Usage
fw.runScript(filename)

Arguments
filename  The name of the script file to execute. If filename is not a file URL (that is, if it does not begin with "file:///"), it is assumed to be the name of a file in the Fireworks /Configuration/Commands folder.

Returns
Result of script.

Description
Executes a JavaScript file.

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.

Usage
fw.saveAll()

Arguments
None.
Returns
Nothing.

Description
Saves all open documents, displaying the Save As dialog box for any documents that were not previously saved.

\texttt{fw.saveAs()}

Availability
Fireworks 10.

Usage
\texttt{fw.saveAs(doc, url, defaultfromoptions)}

Arguments
\texttt{doc} The document object (for example, \texttt{fw.documents[2]}) that specifies the document to be saved. If document is null, the active document is saved.

\texttt{url} The file URL for the saved document, or null if the dialog box is to come up.

\texttt{defaultfromoptions} By default \texttt{false}. Set \texttt{true} to save the document as per the current optimization settings. The extension of the URL changes as per the optimization settings.

Returns
Nothing.

Description
Saves the specified document in the specified filename and format.

\texttt{fw.saveDocument()}

Availability
Fireworks 3.

Usage
\texttt{fw.saveDocument(document, \{fileURL\})}

Arguments
\texttt{document} A Document object, for example, \texttt{fw.documents[2]}, that specifies the document to be saved. If \texttt{document} is null, the active document is saved.

\texttt{fileURL} The name of the saved document, which is expressed as \texttt{file://URL}. This argument is optional. If \texttt{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.
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()`.

See also
`fw.exportDocumentAs()`

`fw.saveDocumentAs()`

Availability
Fireworks 3.

Usage
`fw.saveDocumentAs(document)`

Arguments
document 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.

Description
Displays the Save As dialog box for the specified document, so that 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()`.

See also
`fw.exportDocumentAs()`

`fw.saveDocumentAsTemplate(arg1,arg2)`

Availability
Fireworks CS5.

Usage
`fw.saveDocumentAsTemplate(fw.documents[1],"file:///C:/file2.png");`

Arguments
Arg1 A Document object.
Arg2 A Absolute file path for saving the file as a template.

Returns
None.

Description
Saves the current document as a template.
See also
fw.saveDocumentAsTemplateAsync(arg1, arg2)

fw.saveDocumentAsTemplateAsync(arg1, arg2)

Availability
Fireworks CS5.

Usage
fw.saveDocumentAsTemplateAsync(fw.documents[1], "file:///C:/file2.png");

Arguments
Arg1  A Document object.
Arg2  A Absolute file path for saving the file as a template.

Returns
None.

Description
Works the same as fw.saveDocumentAsTemplate() except for the async mode of save operation.

See also
fw.saveDocumentAsTemplate(arg1, arg2)

fw.saveDocumentCopyAs()

Availability
Fireworks 3.

Usage
fw.saveDocumentCopyAs(document, fileURL)

Arguments
document  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  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.

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().
See also
fw.exportDocumentAs()

fw.saveJsCommand()

Availability
Fireworks 3.

Usage
fw.saveJsCommand(jscode, filename)

Arguments
jscode  The string of code to be saved as a JSF command file.
filename  The name under which the file should be saved.

Returns
Nothing.

Description
Saves the specified string of JavaScript code as a JSF command file. Fireworks writes the file using UTF-8 encoding and prepends the UTF-8 byte order mark to the file that is written.

fw setActiveViewScale()

Availability
Fireworks MX.

Usage
fw.setActiveViewScale(scale, center)

Arguments
scale  A floating-point number where 1.0 is 100%, or normal view, and 1.5 is 150%. Default is 6%.
center  A point that defines the location in the document at which the view should be centered. This argument can be used to navigate around different parts of the document.

Returns
Nothing.

Description
Sets the zoom amount and the center of the view for the current document.

fw setActiveWindow()
Usage
fw.setActiveWindow(document, {trueFalse})

Arguments
document A Document object, for example, fw.documents[2], that specifies which document should be made active.

trueFalse This optional argument is ignored by Fireworks. It is included only for Dreamweaver compatibility.

Returns
Nothing.

Description
Sets the specified document as the active document.

Example
The following command makes the fourth document the active document.
fw.setActiveWindow(fw.documents[3]);

fw.setFloaterGrouping()

Availability
Fireworks 3.

Usage
fw.setFloaterGrouping(panelNameToMove, panelNameToReceive)

Arguments
panelNameToMove A lowercase string that specifies the panel to be moved.
panelNameToReceive A lowercase string that specifies the panel into which the panelNameToMove panel should move. If panelNameToReceive is null, the panelNameToMove panel moves into its own panel. Acceptable values are "find", "project log", "object", "info", "url", "effect", "history", "mixer", "fill", "stroke", "swatches", "layers", "frames", "behaviors", "optimize", "library", "styles", "optimized colors", "options", and "toolbox".

Note: Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are "outside" the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be "mypanel".

Returns
Nothing.

Description
Moves the specified panel into another 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.
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.

```javascript
fw.setFloaterGrouping("stroke", "object");
```

**fw.setFloaterPosition()**

**Availability**
Fireworks 3.

**Usage**
`fw.setFloaterPosition(panelName, boundingRectangle)`

**Arguments**

**Note:** Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are “outside” the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be "mypanel".

- `boundingRectangle`: A rectangle that specifies the size of the panel (see “Rectangle data type” on page 9). Some panels ignore the specified size but place the upper-left corner of the panel at the upper-left of the specified rectangle.

**Returns**
Nothing.

**Description**
Sets the position and size of a panel.

**fw.setFloaterVisibility()**

**Availability**
Fireworks 3.

**Usage**
`fw.setFloaterVisibility(panelName, bVisible)`

**Arguments**
Note: Any panels not specified in the list of valid arguments (such as those in the Command Panels folder, which are “outside” the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is "Align", and a valid name for a custom panel file mypanel.swf would be "mypanel".

**bVisible** If `bVisible` is `true`, the specified panel is visible. If `bVisible` is `false`, the panel is hidden.

Returns
Nothing.

Description
Shows or hides the specified panel.

**fw.setHideAllFloaters()**

Availability
Fireworks 3.

Usage
fw.setHideAllFloaters(bHide)

Arguments
bHide  If `bHide` is `true`, the panels are hidden. If `bHide` is `false`, the panels are visible.

Returns
Nothing.

Description
Shows or hides the panels. This behavior is the same as the Tab key behavior.

**fw.setPref()**

Availability
Fireworks 3.

Usage
fw.setPref(prefname, prefval)

Arguments
prefname and 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.
Description
Sets the value that is associated with the specified Preference key.

See also
“fw.getPref()” on page 219

fw.setUpFindReplace()

Availability
Fireworks 3.

Usage
fw.setUpFindReplace(findSpec)

Arguments
findSpec A Find object (see “Find object” on page 20).

Returns
Nothing.

Description
Sets up a search.

fw.toggleFloater()

Availability
Fireworks 3.

Usage
fw.toggleFloater(panelName)

Arguments
panelName Acceptable values are "find", "project log", "object", "info", "url", "effect", "history", "mixer", "fill", "stroke", "swatches", "layers", "frames", "behaviors", "optimize", "library", "styles", "optimized colors", "options", and "toolbox".

Note: Any panels not specified in the list of valid arguments (like those in the Command Panels folder which are “outside” the Fireworks application) should be named exactly as they appear in the file system without their file extension. For example, the valid argument name for the Align panel (Align.swf) is “Align”, and a valid name for a custom panel file mypanel.swf would be “mypanel”.

Returns
Nothing.
Description
Shows or hides the specified panel, or makes it topmost.

- If the panel is hidden, this function shows it and makes it topmost.
- If the panel is topmost, this function hides it.
- If the panel is shown but is not topmost, this function makes it topmost.

**fw.ungroupPrimitives()**

*Availability*
Fireworks 4.

*Usage*
fw.ungroupPrimitives()

*Arguments*
None.

*Returns*
Nothing.

*Description*
Replaces selected primitive objects with their equivalent paths. The new objects have all the attributes (mask, stroke, fill, and so on) of the replaced ones.

*See also*
dom.addNewRectanglePrimitive()

**fw.updateHTML()**

*Availability*
Fireworks 4.

*Usage*
fw.updateHTML(doc, htmlUrl, bRecoverFromError)

*Arguments*

- **doc** A Document object that specifies the document to be used for updating the HTML (see “The Document object” on page 28). If doc is null, the active document is used.
- **htmlUrl** The filename of 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.
bRecoverFromError 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 A Boolean value: true if the HTML was updated; false otherwise.

Description Updates the HTML that was previously exported from Fireworks.

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.

Usage
```
fw.writePanelStateToFile(filepath)
```

Arguments
filepath A string that identifies the destination XML file in the format file://URL.

Returns Nothing.

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.

fw.yesNoDialog()

Availability Fireworks MX.

Usage
```
fw.yesNoDialog(promptString)
```

Arguments
promptString The prompt message that appears in the dialog box.
Returns
A Boolean value: true if the user selected the Yes button; false otherwise.

Description
Displays a dialog box that contains buttons labeled Yes and No.

Example
The following code displays a dialog box with Yes and No buttons and the message “Would you like to duplicate the element?”

```javascript
var shouldDuplicate = fw.yesNoDialog("Would you like to duplicate the element?");
```
Chapter 5: Objects within Fireworks documents

This chapter describes the objects that can get or set the properties of elements in 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 7.

Note: For information on how to format nonstandard data types, such as rectangle or point, see “Formatting nonstandard data types” on page 8.

Behavior object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>string</td>
<td>The JavaScript call for the behavior. For legal values, see “Using the dom.addBehavior() function” on page 32.</td>
</tr>
<tr>
<td>event</td>
<td>string</td>
<td>Acceptable values are &quot;onMouseOver&quot;, &quot;onClick&quot;, &quot;onMouseOut&quot;, &quot;onLoad&quot;, and &quot;<strong>ANY</strong>&quot; (the <strong>ANY</strong> argument is used as a wildcard value in some situations).</td>
</tr>
</tbody>
</table>

Brush object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>alphaRemap</td>
<td>string</td>
<td>Acceptable values are &quot;none&quot;, &quot;white neon&quot;, &quot;harsh wet&quot;, &quot;smooth neon&quot;, &quot;wavy gravy&quot;, and &quot;white neon edge&quot;.</td>
</tr>
<tr>
<td>angle</td>
<td>integer</td>
<td>0 to 360</td>
</tr>
<tr>
<td>antiAliased</td>
<td>Boolean</td>
<td>If set to true, the brush edges are anti-aliased.</td>
</tr>
<tr>
<td>aspect</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>blackness</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>category</td>
<td>string</td>
<td>Determines in which subsection of the Stroke panel the brush will appear (for example, Pencil, Airbrush, and so on).</td>
</tr>
<tr>
<td>concentration</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>dashOffSize1, dashOffSize2, dashOffSize3</td>
<td>integer</td>
<td>The lengths in pixels of spaces for a dotted line, these values control the first, second, and third spaces, respectively.</td>
</tr>
</tbody>
</table>
### Objects within Fireworks documents

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashOnSize1, dashOnSize2, dashOnSize3</td>
<td>integer</td>
<td>The lengths, in pixels, of dashes for a dotted line, these values control the first, second, and third dashes, respectively.</td>
</tr>
<tr>
<td>diameter</td>
<td>integer</td>
<td>0 to 1000</td>
</tr>
<tr>
<td>feedback</td>
<td>string</td>
<td>Acceptable values are &quot;none&quot;, &quot;brush&quot;, and &quot;background&quot;.</td>
</tr>
<tr>
<td>flowRate</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>maxCount</td>
<td>integer</td>
<td>0 to 64</td>
</tr>
<tr>
<td>minSize</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name of the brush, which is visible in the Stroke panel.</td>
</tr>
<tr>
<td>numDashes</td>
<td>integer</td>
<td>0 to 3</td>
</tr>
<tr>
<td>sense_hdir_angle</td>
<td>float</td>
<td>The <code>sense_</code> properties map directly to the values on the Stroke Options &gt; Advanced dialog &gt; Sensitivity tab (accessible through the Brush property inspector stroke settings); where hdir is the horizontal value and vdir is the vertical value, and blackness is the build-up of black pixels as some tools brush over the same spot repeatedly (like the felt tip).</td>
</tr>
<tr>
<td>sense_hdir_blackness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_hue</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_lightness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_opacity</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_saturation</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_scatter</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_hdir_size</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_angle</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_blackness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_hue</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_lightness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_opacity</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_saturation</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_scatter</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_pressure_size</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_angle</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_blackness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_hue</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_lightness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_opacity</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_saturation</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_random_scatter</td>
<td>float</td>
<td></td>
</tr>
</tbody>
</table>

Last updated 5/17/2012
### EXTENDING FIREWORKS

**Objects within Fireworks documents**

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sense_random_size</td>
<td>float</td>
<td>0 to 100, where x is a value of pressure, speed, hDir, vDir, or random, and y is a value of: size, angle, opacity, blackness, scatter, hue, lightness, or saturation. For example, sensitivity_pressure_size.</td>
</tr>
<tr>
<td>sense_speed_angle</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_blackness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_hue</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_lightness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_opacity</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_saturation</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_scatter</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_speed_size</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_angle</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_blackness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_hue</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_lightness</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_opacity</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_saturation</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_scatter</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sense_vdir_size</td>
<td>float</td>
<td></td>
</tr>
<tr>
<td>sensitivity_x_y</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>shape</td>
<td>string</td>
<td>Acceptable values are &quot;circle&quot; and &quot;square&quot;.</td>
</tr>
<tr>
<td>softenMode</td>
<td>string</td>
<td>Acceptable values are &quot;bell curve&quot; and &quot;linear&quot;.</td>
</tr>
<tr>
<td>softness</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>spacing</td>
<td>float</td>
<td>0 to 500 (a percentage, as much as 500 percent)</td>
</tr>
<tr>
<td>textureBlend</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>textureEdge</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>tipColoringMode</td>
<td>string</td>
<td>Acceptable values are &quot;random&quot;, &quot;uniform&quot;, &quot;complementary&quot;, &quot;hue&quot;, and &quot;shadow&quot;.</td>
</tr>
<tr>
<td>tipCount</td>
<td>integer</td>
<td>1 to 32</td>
</tr>
<tr>
<td>tipSpacing</td>
<td>float</td>
<td>0 to 100</td>
</tr>
<tr>
<td>tipSpacingMode</td>
<td>string</td>
<td>Acceptable values are &quot;random&quot;, &quot;diagonal&quot;, and &quot;circular&quot;.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Acceptable values are &quot;natural&quot; and &quot;simple&quot;.</td>
</tr>
</tbody>
</table>
Contour object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>isClosed</td>
<td>Boolean</td>
<td>If set to true, the path is closed by connecting the final point in the contour with the first point.</td>
</tr>
<tr>
<td>nodes</td>
<td>array</td>
<td>Array of ContourNode objects on the contour (for more information, see “ContourNode object” on page 246).</td>
</tr>
</tbody>
</table>

ContourNode object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamicInfo</td>
<td>array</td>
<td>Array of ContourNodeDynamicInfo objects on this ContourNode object (for more information, see “ContourNodeDynamicInfo object” on page 247).</td>
</tr>
<tr>
<td>isCurvePoint</td>
<td>Boolean</td>
<td>If set to true, this point’s control points are constrained to be linear with the main point, which forces a smooth curve. If set to false, there are no constraints on the control points.</td>
</tr>
<tr>
<td>isSelectedPoint</td>
<td>Boolean</td>
<td>If set to true, this point was subselected (for example, by the subselection tool).</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>A unique name assigned to the object.</td>
</tr>
<tr>
<td>predX</td>
<td>float</td>
<td>The x coordinate of the contour node’s preceding control point.</td>
</tr>
<tr>
<td>predY</td>
<td>float</td>
<td>The y coordinate of the contour node’s preceding control point.</td>
</tr>
<tr>
<td>randomSeed</td>
<td>integer</td>
<td>0 to 65,535</td>
</tr>
<tr>
<td>succX</td>
<td>float</td>
<td>The x coordinate of the contour node’s following control point.</td>
</tr>
<tr>
<td>succY</td>
<td>float</td>
<td>The y coordinate of the contour node’s following control point.</td>
</tr>
<tr>
<td>x</td>
<td>float</td>
<td>The x coordinate of the contour node’s main control point.</td>
</tr>
<tr>
<td>y</td>
<td>float</td>
<td>The y coordinate of the contour node’s main control point.</td>
</tr>
</tbody>
</table>

The following table lists the methods of the ContourNode object, along with their parameters.
ContourNodeDynamicInfo object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>float</td>
<td>0.0 to 65,535.0 milliseconds</td>
</tr>
<tr>
<td>pressure</td>
<td>float</td>
<td>0.0 to 1.0</td>
</tr>
<tr>
<td>velocity</td>
<td>float</td>
<td>0.0 to 255.9999 pixels per millisecond</td>
</tr>
</tbody>
</table>

ControlPoint object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>hiliteDragOverObject</td>
<td>Boolean</td>
<td>If true, Fireworks highlights an object when a control point is dragged over it.</td>
</tr>
<tr>
<td>index</td>
<td>integer</td>
<td>Index for the control point.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Assigned name of the control point.</td>
</tr>
<tr>
<td>toolTip</td>
<td>string</td>
<td>Text to display when the user rolls the pointer (mouse) over the control point.</td>
</tr>
</tbody>
</table>
The following table lists the methods of the ControlPoint object, along with their parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisterMove()</td>
<td>object</td>
<td>The RegisterMoveParms object containing the move parameters. Use smartShape.GetDefaultMoveParms() to obtain this object, then adjust properties as needed. For a list of properties, see “RegisterMoveParms object” on page 269.</td>
</tr>
<tr>
<td>RegisterLinearMove()</td>
<td>point</td>
<td>A point, which in combination with the node point, defines the line to move along.</td>
</tr>
<tr>
<td></td>
<td>object</td>
<td>The RegisterMoveParms object containing the move parameters. Use smartShape.GetDefaultMoveParms() to obtain this object, then adjust properties as needed. For a list of properties, see “RegisterMoveParms object” on page 269.</td>
</tr>
<tr>
<td>RegisterCircularMove()</td>
<td>point</td>
<td>The center point for the circular movement.</td>
</tr>
<tr>
<td></td>
<td>object</td>
<td>The RegisterMoveParms object containing the move parameters. Use smartShape.GetDefaultMoveParms() to obtain this object, then adjust properties as needed. For a list of properties, see “RegisterMoveParms object” on page 269.</td>
</tr>
<tr>
<td>RegisterPolygonMove()</td>
<td>point</td>
<td>The center point for the polygon.</td>
</tr>
<tr>
<td></td>
<td>object</td>
<td>The RegisterMoveParms object containing the move parameters. Use smartShape.GetDefaultMoveParms() to obtain this object, then adjust properties as needed. For a list of properties, see “RegisterMoveParms object” on page 269.</td>
</tr>
<tr>
<td>RegisterInsertBBoxMove()</td>
<td>object</td>
<td>The RegisterMoveParms object containing the move parameters. Use smartShape.GetDefaultMoveParms() to obtain this object, then adjust properties as needed. For a list of properties, see “RegisterMoveParms object” on page 269.</td>
</tr>
</tbody>
</table>

**Effect object**

Each Fireworks Effect (bevel, drop shadow, etc.) has a unique set of attributes. So, each Effect object has its own set of properties that can be set (instead of a common set of properties for all Effect objects). 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 object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AngleSoftness</td>
<td>integer</td>
<td>Specifies the blur, or feather amount, for the shadow and highlight colors of the bevel.</td>
</tr>
<tr>
<td>BevelContrast</td>
<td>integer</td>
<td>0 to 100 percent</td>
</tr>
<tr>
<td>BevelType</td>
<td>integer</td>
<td>Sets a bevel as inner, outer, raised embossed, inset embossed, or glow effect, as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>InnerBevel = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OuterBevel = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RaiseEmboss = 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>InsetEmboss = 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GlowEffect = 4</td>
</tr>
<tr>
<td>BevelWidth</td>
<td>integer</td>
<td>The width of the bevel, in pixels.</td>
</tr>
<tr>
<td>ButtonState</td>
<td>integer</td>
<td>BevelButtonUp = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BevelButtonOver = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BevelButtonDown = 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BevelButtonHit = 3</td>
</tr>
<tr>
<td>DownBlendColor</td>
<td>string</td>
<td>A color string that specifies the color that is blended on top of the image if ButtonState = 2 (BevelButtonDown) (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>EdgeThreshold</td>
<td>integer</td>
<td>Controls the opacity at which the edge of the effect is defined. Use 1 if BevelType = 4 (for GlowEffect); otherwise, use 0.</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{7fe61102-6cee-11d1-8c76000502701850}&quot;</td>
</tr>
<tr>
<td>EmbossFaceColor</td>
<td>string</td>
<td>A color string that specifies the color that is blended onto the face of the object when embossing (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>GlowStartDistance</td>
<td>integer</td>
<td>Specifies how far away from the object the glow starts, in pixels. Specify a negative value to create “ring” glows and a positive value to create “halo” glows.</td>
</tr>
<tr>
<td>GlowWidth</td>
<td>integer</td>
<td>The width of the glow, in pixels.</td>
</tr>
<tr>
<td>HiliteColor</td>
<td>string</td>
<td>A color string that specifies the color that is blended to provide the spectral lighting type effect (for more information, see “Color string data type” on page 8). Used by beveling only. Currently white is always used for internally created effects (although any value should work). This is the complement of ShadowColor.</td>
</tr>
<tr>
<td>HitBlendColor</td>
<td>string</td>
<td>A color string that specifies the color that is blended on the face of the image if ButtonState = 3 (BevelButtonHit) (for more information, see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>LightAngle</td>
<td>integer</td>
<td>The light angle, in degrees, that is used to create the light and shadow effects for the bevel.</td>
</tr>
<tr>
<td>MaskSoftness</td>
<td>integer</td>
<td>The feather amount on the glow edge, in pixels.</td>
</tr>
</tbody>
</table>
### OuterBevelColor

**Data type**: string  
*A color string that specifies the color of the outer bevel effect (for more information, see “Color string data type” on page 8).*

### ShadowColor

**Data type**: string  
*A color string that specifies the color that is blended to provide the bevel shadow effect (for more information, see “Color string data type” on page 8). Currently black is always used for internally created effects (though any value should work). This is the complement of HiliteColor.*

### ShowObject

**Data type**: Boolean  
The default value is false.

### SlopeMultiplier

**Data type**: 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

**Data type**: 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 object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td><code>{f1cfce41-718e-11d1-8c8200a024cdc039}</code></td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
</tbody>
</table>

#### Blur More object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td><code>{f1cfce42-718e-11d1-8c8200a024cdc039}</code></td>
</tr>
</tbody>
</table>

#### Brightness/Contrast object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>brightness_amount</td>
<td>integer</td>
<td>-100 to 100</td>
</tr>
<tr>
<td>contrast_amount</td>
<td>integer</td>
<td>-100 to 100</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td><code>{3439b08c-1921-11d1-9bde00e02910d680}</code></td>
</tr>
</tbody>
</table>
Convert to Alpha object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;[2932d5a2-ca48-11d1-8561000502701850]&quot;</td>
</tr>
</tbody>
</table>

Curves object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;[3439b08e-1923-11d3&quot;-9bde00e02910d80]&quot;</td>
</tr>
<tr>
<td>rgb_points</td>
<td>vector of points</td>
<td>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 the points’ x coordinate values.</td>
</tr>
<tr>
<td>red_points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>green_points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blue_points</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Drop Shadow object

You can find the Filter options for objects in the Properties panel. Select the object you want to have a shadow, then look in Filters > Shadow and Glow > Drop Shadow for the options.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;[a7944db8-6ce2-11d1-8c7600a024cdc039]&quot;</td>
</tr>
<tr>
<td>ShadowAngle</td>
<td>float</td>
<td>The angle of the shadow, in degrees.</td>
</tr>
<tr>
<td>ShadowBlur</td>
<td>integer</td>
<td>The feathering amount of the shadow edges, in pixels.</td>
</tr>
<tr>
<td>ShadowColor</td>
<td>string</td>
<td>A color string that specifies the color of the shadow (for more information, see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>ShadowDistance</td>
<td>integer</td>
<td>The offset of the shadow, in pixels.</td>
</tr>
<tr>
<td>ShadowType</td>
<td>integer</td>
<td>0 = normal shadow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = knockout shadow</td>
</tr>
</tbody>
</table>

Find Edges object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;[fc7093f1-f95c-11d0-8be200a024cdc039]&quot;</td>
</tr>
</tbody>
</table>
## Gaussian Blur object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{d04ef8c0-71b3-11d1-8c8200a024cdc039}&quot;</td>
</tr>
<tr>
<td>gaussian_blur_radius</td>
<td>float</td>
<td>0.1 to 250</td>
</tr>
</tbody>
</table>

## Hue/Saturation object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{3439b08d-1922-11d3-9bde00e02910d580}&quot;</td>
</tr>
<tr>
<td>hue_amount</td>
<td>integer</td>
<td>-180 to 180 if hls_colorize is false; 0 to 360 if hls_colorize is true.</td>
</tr>
<tr>
<td>saturation_amount</td>
<td>integer</td>
<td>-100 to 100 if hls_colorize is false; 0 to 100 if hls_colorize is true.</td>
</tr>
<tr>
<td>lightness_amount</td>
<td>integer</td>
<td>0 to 100</td>
</tr>
<tr>
<td>hls_colorize</td>
<td>Boolean</td>
<td>Specifies whether the effect should automatically colorize. Default value is false.</td>
</tr>
</tbody>
</table>

## Inner Shadow object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{5600f702-774c-11d3-baad0000861f4d01}&quot;</td>
</tr>
<tr>
<td>ShadowAngle</td>
<td>integer</td>
<td>The angle of the shadow, in degrees.</td>
</tr>
<tr>
<td>ShadowBlur</td>
<td>integer</td>
<td>The feathering amount of the shadow edges, in pixels.</td>
</tr>
<tr>
<td>ShadowColor</td>
<td>string</td>
<td>A color string that specifies the color of the shadow (for more information, see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>ShadowDistance</td>
<td>integer</td>
<td>The offset of the shadow, in pixels.</td>
</tr>
<tr>
<td>ShadowType</td>
<td>integer</td>
<td>0 = normal shadow \n 1 = knockout shadow</td>
</tr>
</tbody>
</table>

## Invert object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{d2541291-70d6-11d1-8c8200a024cdc039}&quot;</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
</tbody>
</table>
Levels object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{d04ef8c1-71b4-11d1-8c8200a024cdc039}&quot;</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>source_low_rgb*</td>
<td>integer</td>
<td>These source* values are all input levels to the filter, with values of 0 to 255.</td>
</tr>
<tr>
<td>source_high_rgb*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_low_red*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_high_red*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_low_green*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_high_green*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_low_blue*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_high_blue*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_low_rgb</td>
<td>integer</td>
<td>These dest* values are all output levels to the filter, with values of 0 to 255.</td>
</tr>
<tr>
<td>dest_high_rgb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_low_red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_high_red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_low_green</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_high_green</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_low_blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dest_high_blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gamma_rgb</td>
<td>float</td>
<td>These gamma* values are all gamma levels to the filter, with values of 0.1 to 10.0.</td>
</tr>
<tr>
<td>gamma_red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gamma_green</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gamma_blue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sharpen object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{c20952b1-fc76-11d0-8be700a024cdc039}&quot;</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
</tbody>
</table>
Sharpen More object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{1f2f2591-9db7-11d1-8cad00a024cdc039}&quot;</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
</tbody>
</table>

Unsharp Mask object

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EffectMoaID</td>
<td>string</td>
<td>&quot;{f1cfcce44-718e-11d1-8c8200a024cdc039}&quot;</td>
</tr>
<tr>
<td>EffectIsVisible</td>
<td>Boolean</td>
<td>If set to false, the effect is included but temporarily hidden. The default value is true.</td>
</tr>
<tr>
<td>unsharp_mask_amount</td>
<td>integer</td>
<td>1 to 500</td>
</tr>
<tr>
<td>unsharp_mask_radius</td>
<td>float</td>
<td>0.1 to 250</td>
</tr>
<tr>
<td>unsharp_mask_threshold</td>
<td>integer</td>
<td>0 to 255</td>
</tr>
</tbody>
</table>

EffectList object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>string</td>
<td>Specifies which subheading in the Effects panel to use.</td>
</tr>
<tr>
<td>effects</td>
<td>array</td>
<td>Array of Effect objects (for more information, see &quot;Effect object&quot; on page 248).</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name that appears in the Effects panel.</td>
</tr>
</tbody>
</table>

Element object

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 (•).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>blendMode</td>
<td>string</td>
<td>Acceptable values are &quot;normal&quot;, &quot;multiply&quot;, &quot;screen&quot;, &quot;darken&quot;, &quot;lighten&quot;, &quot;difference&quot;, &quot;hue&quot;, &quot;saturation&quot;, &quot;color&quot;, &quot;luminosity&quot;, &quot;invert&quot;, &quot;tint&quot;, and &quot;erase&quot;.</td>
</tr>
<tr>
<td>customData</td>
<td>struct</td>
<td>Assign any objects (array, integer, string, and so on).</td>
</tr>
<tr>
<td>effectList</td>
<td>object</td>
<td>EffectList object (for more information, see &quot;EffectList object&quot; on page 254).</td>
</tr>
</tbody>
</table>
## Objects within Fireworks documents

### Group object

Group is a subclass of the base class Element and contains the following properties in addition to those in Element (for more information, see “Element object” on page 254).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlPoints</td>
<td>array</td>
<td>Array of control points defined for the Auto Shape object.</td>
</tr>
<tr>
<td>elements</td>
<td>array</td>
<td>Array of Element objects in the group (for more information, see “Element object” on page 254).</td>
</tr>
</tbody>
</table>

---

The following table lists the methods of the Element object, along with their parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateSmartShapeCode</td>
<td>root</td>
<td>The root parameter is a string value that is prefixed to each line of output.</td>
</tr>
</tbody>
</table>

---

Last updated 5/17/2012
The following table lists the methods of the Group object, along with their parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>generateSmartShapeCode()</td>
<td>string</td>
<td>Generates JavaScript code for creating an Auto Shape. You can specify a string to prefix each line of output.</td>
</tr>
<tr>
<td>globalToSmartShapeCoords()</td>
<td>point</td>
<td>Changes an object to the new coordinates after an Auto Shape is transformed (scaled, skewed, or rotated) so that the Auto Shape object contains the new location.</td>
</tr>
<tr>
<td>RegisterForEvent()</td>
<td>string</td>
<td>Call this to receive notification of the string specifying a Fireworks event. smartShape.operation will be the name of the event triggered. Returns the total number of events registered after adding the specified event.</td>
</tr>
<tr>
<td>removeTransformation()</td>
<td>none</td>
<td>Undoes the previous transformation.</td>
</tr>
<tr>
<td>smartShapeToGlobalCoords()</td>
<td>point</td>
<td>Converts a transformed (scaled, skewed, or rotated) SmartShape object’s space into global space. (For more information, see globalToSmartShapeCoords() above.)</td>
</tr>
<tr>
<td>unRegisterAllEvents()</td>
<td>none</td>
<td>Call this to stop receiving notification of all previously registered events.</td>
</tr>
<tr>
<td>UnRegisterForEvent()</td>
<td>string</td>
<td>Call this to stop receiving notification of a single previously registered event.</td>
</tr>
</tbody>
</table>

**Image object**

Image is a subclass of the base class Element (for more information, see “Element object” on page 254). It contains no properties or methods other than those in Element.

**Instance object**

Instance is a subclass of the base class Element and contains the following properties in addition to those in Element (for more information, see “Element object” on page 254). Read-only properties are marked with a bullet (•).
Objects within Fireworks documents

Hotspot object

A Hotspot converts to 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 (for more information, see “Element object” on page 254).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altText</td>
<td>string</td>
<td>The alternate text description.</td>
</tr>
<tr>
<td>instanceType</td>
<td>string</td>
<td>The type of element, for example &quot;graphic&quot;, &quot;button&quot;, or &quot;animation&quot;.</td>
</tr>
<tr>
<td>symbolID</td>
<td>string</td>
<td>An arbitrary string that uniquely identifies the symbol that owns this instance.</td>
</tr>
<tr>
<td>symbolName</td>
<td>string</td>
<td>Specifies the symbol name.</td>
</tr>
<tr>
<td>targetText</td>
<td>string</td>
<td>The target.</td>
</tr>
<tr>
<td>transform</td>
<td>Matrix</td>
<td>Specifies the transformation matrix of a symbol instance.</td>
</tr>
<tr>
<td>transformMode</td>
<td>string</td>
<td>Acceptable values are &quot;paths&quot; and &quot;pixels&quot;.</td>
</tr>
<tr>
<td>urlText</td>
<td>string</td>
<td>The link text.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altText</td>
<td>string</td>
<td>Text that is written into the HTML Alt tag when exporting.</td>
</tr>
<tr>
<td>behaviors</td>
<td>array</td>
<td>Array of Behavior objects for the Hotspot (for more information, see “Behavior object” on page 243).</td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td>Color in which the Hotspot is drawn in the Document window. Default value is &quot;#00FFFF&quot;.</td>
</tr>
<tr>
<td>contour</td>
<td>object</td>
<td>Contour object for the Hotspot (for more information, see “Contour object” on page 246). Used only if shape=&quot;polyline&quot;; otherwise null.</td>
</tr>
<tr>
<td>shape</td>
<td>string</td>
<td>Acceptable values are &quot;rectangle&quot;, &quot;circle&quot;, and &quot;polyline&quot;.</td>
</tr>
<tr>
<td>targetText</td>
<td>string</td>
<td>Text that is written into the HTML Target tag when exporting.</td>
</tr>
<tr>
<td>urlText</td>
<td>string</td>
<td>Text that is written into the HTML Href tag when exporting.</td>
</tr>
</tbody>
</table>

SliceHotspot object

A slice Hotspot converts to 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 (for more information, see “Hotspot object” on page 257). Read-only properties are marked with a bullet (•).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseName</td>
<td>string</td>
<td>Base name for slice filenames, or null for automatic name.</td>
</tr>
<tr>
<td>exportOptions</td>
<td>object</td>
<td>ExportOptions object (for more information, see “ExportOptions object” on page 260); null if using current document defaults.</td>
</tr>
<tr>
<td>htmlText</td>
<td>string</td>
<td>If slice kind is set to &quot;empty&quot;, this text is exported instead of the image. The default is an empty string.</td>
</tr>
</tbody>
</table>
### Path object

Path is a subclass of the base class Element and contains the following properties in addition to those in Element (for more information, see “Element object” on page 254).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sliceID</td>
<td>string</td>
<td>An arbitrary string that uniquely identifies this slice.</td>
</tr>
<tr>
<td>sliceKind</td>
<td>string</td>
<td>If set to &quot;image&quot;, generates an image; if set to &quot;empty&quot;, generates the text specified by htmlText.</td>
</tr>
<tr>
<td>tdTagText</td>
<td>string</td>
<td>This string contains all the attributes of a table cell except the colspan and rowspan values. An example value is &quot;bgcolor=ff0000&quot; valign=&quot;top&quot;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>contours</td>
<td>array</td>
<td>Array of Contour objects on this Path object (for more information, see “Contour object” on page 246).</td>
</tr>
<tr>
<td>isEvenOddFill</td>
<td>Boolean</td>
<td>true if the path uses an even/odd fill.</td>
</tr>
<tr>
<td>pathAttributes</td>
<td>object</td>
<td>PathAttrs object (for more information, see “PathAttrs object” on page 268).</td>
</tr>
<tr>
<td>randSeed</td>
<td>float</td>
<td>A 32-bit integer. JavaScript integers hold only 31-bit numbers, so it is stored as a floating-point number.</td>
</tr>
<tr>
<td>textureOffset</td>
<td>point</td>
<td>If the path has a textured brush or fill, specifies the offset of the texture’s origin.</td>
</tr>
<tr>
<td>pathOperation</td>
<td>String</td>
<td>Defines the path operation used for a sub-element within a CompoundShape.</td>
</tr>
</tbody>
</table>

### Text object

Text is a subclass of the base class Element and contains the following properties in addition to those in Element (for more information, see “Element object” on page 254).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>antiAliased</td>
<td>Boolean</td>
<td>If set to true (the default), anti-aliases the text.</td>
</tr>
<tr>
<td>antiAliasMode</td>
<td>string</td>
<td>Acceptable values are &quot;smooth&quot;, &quot;crisp&quot;, and &quot;strong&quot;. This value is ignored if the antiAliased property is set to false.</td>
</tr>
<tr>
<td>autoExpand</td>
<td>Boolean</td>
<td>If set to true, the bounding box will expand automatically to fit a line of text to prevent word wrapping.</td>
</tr>
<tr>
<td>autoKern</td>
<td>Boolean</td>
<td>If set to true, uses pair-kerning information in the fonts to kern the text. If set to false, pair-kerning information in the fonts is ignored. Default value is true.</td>
</tr>
<tr>
<td>orientation</td>
<td>string</td>
<td>Acceptable values are &quot;horizontal left to right&quot; (the default), &quot;vertical right to left&quot;, &quot;horizontal right to left&quot;, and &quot;vertical left to right&quot;.</td>
</tr>
<tr>
<td>pathAttributes</td>
<td>object</td>
<td>PathAttrs object (for more information, see “PathAttrs object” on page 268).</td>
</tr>
</tbody>
</table>
Texture object

The Texture object has the following read-only property.

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name that appears in the Brush or Fill panels.</td>
</tr>
</tbody>
</table>

ElementMask object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoExpandImages</td>
<td>Boolean</td>
<td>If set to true, and the element mask is an image, the image is automatically expanded to fill the entire document, with areas &quot;outside&quot; the image showing through. If set to false (or if the element mask is not an image), areas &quot;outside&quot; the element mask are knocked out.</td>
</tr>
<tr>
<td>element</td>
<td>object</td>
<td>Element object (for more information, see &quot;Element object&quot; on page 254).</td>
</tr>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>If set to true, the mask applies to the element. If set to false, the mask remains present but does not visually affect the element in any way. Default value is true.</td>
</tr>
<tr>
<td>linked</td>
<td>Boolean</td>
<td>If set to true, moving the mask moves the element that owns it, and vice versa. If set to 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.</td>
</tr>
</tbody>
</table>
ExportFrameInfo object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>string</td>
<td>Acceptable values are &quot;mask to image&quot; and &quot;mask to path&quot;.</td>
</tr>
<tr>
<td>owner</td>
<td>object</td>
<td>The element (image, path, text, and so on) that owns the mask.</td>
</tr>
<tr>
<td>showAttrs</td>
<td>Boolean</td>
<td>If set to true, and mode is &quot;mask to path&quot;, the mask element's fill and stroke (if any) are drawn. If set to false, the mask element's fill and stroke are ignored.</td>
</tr>
</tbody>
</table>

ExportOptions object

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

Use the following information to understand the rules for determining scaling in this object:

- If useScale is set to true (the default), percentScale is used to uniformly scale the object on export, and applyScale is ignored.
- If useScale is set to false and applyScale is set to false (the default), no scaling is performed on the object when it is exported.
- If useScale is set to false and applyScale is set to true, then xSize and ySize determine scaling as follows:
  - If the value is positive, specifies the exact size for the axis.
  - If the value is zero, specifies that the axis varies without limit.
  - If the value is negative, specifies that the axis varies but can be no larger than \( \text{abs(value)} \).  
- 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
The following table lists the properties of the ExportOptions object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>animAutoCrop</td>
<td>Boolean</td>
<td>The default value is true.</td>
</tr>
<tr>
<td>animAutoDifference</td>
<td>Boolean</td>
<td>The default value is true.</td>
</tr>
<tr>
<td>applyScale</td>
<td>Boolean</td>
<td>The default value is true.</td>
</tr>
<tr>
<td>colorMode</td>
<td>string</td>
<td>Acceptable values are &quot;indexed&quot; (the default), &quot;24 bit&quot;, and &quot;32 bit&quot;.</td>
</tr>
<tr>
<td>crop</td>
<td>Boolean</td>
<td>The default value is false.</td>
</tr>
<tr>
<td>cropBottom</td>
<td>integer</td>
<td>The default value is 0.</td>
</tr>
<tr>
<td>cropLeft</td>
<td>integer</td>
<td>The default value is 0.</td>
</tr>
<tr>
<td>cropRight</td>
<td>integer</td>
<td>The default value is 0.</td>
</tr>
<tr>
<td>cropTop</td>
<td>integer</td>
<td>The default value is 0.</td>
</tr>
<tr>
<td>ditherMode</td>
<td>string</td>
<td>Acceptable values are &quot;none&quot; (the default), &quot;diffusion&quot;, and &quot;2 by 2&quot;.</td>
</tr>
<tr>
<td>ditherPercent</td>
<td>integer</td>
<td>0 to 100; default value is 100.</td>
</tr>
<tr>
<td>exportFormat</td>
<td>string</td>
<td>Acceptable values are &quot;GIF&quot;, &quot;JPEG&quot;, &quot;PNG&quot;, &quot;custom&quot;, and &quot;GIF animation&quot;. There is no default; this value must be specified.</td>
</tr>
<tr>
<td>frameInfo</td>
<td>array</td>
<td>Array of ExportFrameInfo objects (for more information, see &quot;ExportFrameInfo object&quot; on page 260); can be null (the default).</td>
</tr>
<tr>
<td>interlacedGIF</td>
<td>Boolean</td>
<td>The default value is false.</td>
</tr>
<tr>
<td>jpegQuality</td>
<td>integer</td>
<td>1 to 100; the default value is 80.</td>
</tr>
<tr>
<td>jpegSmoothness</td>
<td>integer</td>
<td>0 to 8; the default value is 0.</td>
</tr>
<tr>
<td>jpegSubsampling</td>
<td>integer</td>
<td>0 to 4; the default value is 1.</td>
</tr>
<tr>
<td>localAdaptive</td>
<td>Boolean</td>
<td>The default value is true.</td>
</tr>
<tr>
<td>lossyGifAmount</td>
<td>integer</td>
<td>0 to 100; the default value is 0.</td>
</tr>
<tr>
<td>macFileCreator</td>
<td>string</td>
<td>The default value is &quot;&quot; (an empty string).</td>
</tr>
<tr>
<td>macFileType</td>
<td>string</td>
<td>The default value is &quot;&quot; (an empty string).</td>
</tr>
</tbody>
</table>
The following table lists the properties of the ExportPaletteInfo object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorLocked</td>
<td>Boolean</td>
<td>Set to <code>true</code> if the color is locked in the panel. The default value is <code>false</code>.</td>
</tr>
<tr>
<td>colorModified</td>
<td>Boolean</td>
<td>Set to <code>true</code> if the color was edited. The default value is <code>false</code>.</td>
</tr>
</tbody>
</table>
The following table lists the properties of the ExportSettings object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorSelected</td>
<td>Boolean</td>
<td>Set to true if the color is selected in the panel (selection is a temporary attribute). The default value is false.</td>
</tr>
<tr>
<td>colorTransparent</td>
<td>Boolean</td>
<td>Set to true if the color is exported as transparent. The default value is false.</td>
</tr>
<tr>
<td>newColorValue</td>
<td>string</td>
<td>If colorModified is set to true, specifies the color that will actually be used. The default value is &quot;#000000&quot;.</td>
</tr>
<tr>
<td>baseName</td>
<td>string</td>
<td>The name from which all automatically named slice names are derived.</td>
</tr>
<tr>
<td>discardUnspecifiedSlices</td>
<td>Boolean</td>
<td>If set to true, omits undefined slices from export operations.</td>
</tr>
<tr>
<td>docHtmlEncoding</td>
<td>string</td>
<td>Determines the encoding standard for the HTML file that Fireworks generates during export. Use &quot;iso-8859-1&quot; for ASCII or &quot;utf-8&quot; for Unicode.</td>
</tr>
<tr>
<td>docXHTMLFormat</td>
<td>Boolean</td>
<td>Determines whether Fireworks outputs XHTML formatted files (true) or HTML formatted files (false) when the user exports a file.</td>
</tr>
<tr>
<td>exportFilesStyle</td>
<td>string</td>
<td>Acceptable values are: &quot;HTML and Images&quot; &quot;Images Only&quot; &quot;Dreamweaver LBI&quot; &quot;Director HTML&quot; &quot;CSS Layers&quot; &quot;Layers to Files&quot; &quot;Frames to Files&quot; &quot;Lotus Domino&quot; &quot;Adobe Flash SWF&quot; &quot;Illustrator&quot; &quot;Photoshop&quot;</td>
</tr>
<tr>
<td>fileExtension</td>
<td>string</td>
<td>Defines the extension to append to the filename.</td>
</tr>
<tr>
<td>generateDemoHtml</td>
<td>Boolean</td>
<td>If set to true, generates multiple HTML pages for button export.</td>
</tr>
<tr>
<td>htmlDestination</td>
<td>string</td>
<td>Acceptable values are &quot;same&quot;, &quot;custom&quot;, and &quot;clipboard&quot;.</td>
</tr>
<tr>
<td>setByUser</td>
<td>Boolean</td>
<td>If set to true, the user specifies the export settings. If set to false, the first time the file is exported, Fireworks chooses settings based on the data.</td>
</tr>
<tr>
<td>shimGeneration</td>
<td>string</td>
<td>Acceptable values are &quot;none&quot; (no shims), &quot;transparent&quot; (one-pixel transparent shims), and &quot;nested tables&quot; (no shims, but nested tables).</td>
</tr>
<tr>
<td>sliceAlongGuides</td>
<td>Boolean</td>
<td>If set to true, use guides for slicing (and sliceUsingUrls should be set to false).</td>
</tr>
<tr>
<td>Property</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sliceAutoNaming1</td>
<td>string</td>
<td>Used to generate a name by concatenating six strings. If you need fewer than six strings, fill in the remaining strings with &quot;none&quot;.</td>
</tr>
<tr>
<td>sliceAutoNaming2</td>
<td></td>
<td>Acceptable values are:</td>
</tr>
<tr>
<td>sliceAutoNaming3</td>
<td></td>
<td>*none&quot; — generates nothing.</td>
</tr>
<tr>
<td>sliceAutoNaming4</td>
<td></td>
<td>*row_col&quot; — generates a unique row and column index; 0_0 is first, 0_1 is second, and so on.</td>
</tr>
<tr>
<td>sliceAutoNaming5</td>
<td></td>
<td>*ALPHA&quot; — generates a unique uppercase letter: A is first, B is second, and so on.</td>
</tr>
<tr>
<td>sliceAutoNaming6</td>
<td></td>
<td>*alpha&quot; — generates a unique lowercase letter: a is first, b is second, and so on.</td>
</tr>
<tr>
<td>sliceAutoNaming7</td>
<td></td>
<td>*numeric1&quot; — generates a unique number: 1 is first, 2 is second, and so on.</td>
</tr>
<tr>
<td>sliceAutoNaming8</td>
<td></td>
<td>*numeric01&quot; — generates a unique two-digit number: 01 is first, 02 is second, and so on.</td>
</tr>
<tr>
<td>sliceAutoNaming9</td>
<td></td>
<td>*doc.name&quot; — name of the file being exported, without a path or extension, such as &quot;image&quot;.</td>
</tr>
<tr>
<td>sliceAutoNaming10</td>
<td></td>
<td>*slice&quot; — the string &quot;slice&quot;.</td>
</tr>
<tr>
<td>sliceAutoNaming11</td>
<td></td>
<td>*underscore&quot; — the underscore character (_)</td>
</tr>
<tr>
<td>sliceAutoNaming12</td>
<td></td>
<td>*period&quot; — the period character (.)</td>
</tr>
<tr>
<td>sliceAutoNaming13</td>
<td></td>
<td>*space&quot; — the space character ( )</td>
</tr>
<tr>
<td>sliceAutoNaming14</td>
<td></td>
<td>*hyphen&quot; — the hyphen character (-)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, to generate names of &quot;image_slice01&quot;, &quot;image_slice02&quot;, and so on from a document named &quot;image&quot;, set the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming1: &quot;doc.name&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming2: &quot;underscore&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming3: &quot;slice&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming4: &quot;numeric01&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming5: &quot;none&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sliceAutoNaming6: &quot;none&quot;</td>
</tr>
</tbody>
</table>
The following table lists the properties of the Fill object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>string</td>
<td>Specifies where this fill appears in the Fill panel.</td>
</tr>
<tr>
<td>ditherColors</td>
<td>array</td>
<td>Array of two color strings (see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>edgeType</td>
<td>string</td>
<td>Acceptable values are &quot;hard&quot; and &quot;antialiased&quot;.</td>
</tr>
<tr>
<td>feather</td>
<td>integer</td>
<td>0 to 1000, which represents the feathering value in pixels (0 means no feathering).</td>
</tr>
<tr>
<td>gradient</td>
<td>object</td>
<td>Gradient object (see &quot;Gradient object&quot; on page 267).</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name that appears in the Fill panel.</td>
</tr>
<tr>
<td>pattern</td>
<td>object</td>
<td>Pattern object (see &quot;Pattern object&quot; on page 269).</td>
</tr>
<tr>
<td>shape</td>
<td>string</td>
<td>Acceptable values are &quot;solid&quot;, &quot;linear&quot;, &quot;radial&quot;, &quot;conical&quot;, &quot;satin&quot;, &quot;pinch&quot;, &quot;folds&quot;, &quot;elliptical&quot;, &quot;rectangular&quot;, &quot;bars&quot;, &quot;ripple&quot;, &quot;waves&quot;, &quot;pattern&quot;, and &quot;web dither&quot;.</td>
</tr>
</tbody>
</table>
### Frame object

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 (•).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>delay</td>
<td>integer</td>
<td>Hundredths of a second.</td>
</tr>
<tr>
<td>disposal</td>
<td>string</td>
<td>Acceptable values are &quot;unspecified&quot;, &quot;none&quot;, &quot;background&quot;, and &quot;previous&quot;.</td>
</tr>
<tr>
<td>layers •</td>
<td>array</td>
<td>Array of FrameNLayerIntersect objects in the document (see &quot;FrameNLayerIntersection object&quot; on page 266).</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Specifies the name of the state.</td>
</tr>
<tr>
<td>topLayers</td>
<td>array</td>
<td>Array of top layers returned as FrameNLayerIntersection objects.</td>
</tr>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>If set to false, this frame is hidden. Default value is true.</td>
</tr>
</tbody>
</table>

### FrameNLayerIntersection object

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 (•).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>elements •</td>
<td>array</td>
<td>Array of Element objects (see &quot;Element object&quot; on page 254).</td>
</tr>
<tr>
<td>elementsandsublayers</td>
<td>array</td>
<td>Array of elements and sublayers obtained from topLayers.</td>
</tr>
<tr>
<td>locked</td>
<td>Boolean</td>
<td>If set to true, this FrameNLayerIntersection object is locked. The default value is false.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Specifies the name of the layer when the layers are accessed from states.</td>
</tr>
<tr>
<td>visible</td>
<td>Boolean</td>
<td>If set to false, this FrameNLayerIntersection object is hidden. The default value is true.</td>
</tr>
</tbody>
</table>
Gradient object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name that appears in the Fill panel.</td>
</tr>
<tr>
<td>nodes</td>
<td>array</td>
<td>Array of GradientNode objects (see &quot;GradientNode object&quot; on page 267).</td>
</tr>
<tr>
<td>opacityNodes</td>
<td>array</td>
<td>Array of GradientNode objects (see &quot;GradientNode object&quot; on page 267), that identify the opacity ramp associated with a gradient.</td>
</tr>
</tbody>
</table>

GradientNode object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>string</td>
<td>A color string that specifies the color at this position in the gradient (see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>isOpacityNode</td>
<td>Boolean</td>
<td>If set to true, this node is part of the gradient’s opacity ramp.</td>
</tr>
<tr>
<td>position</td>
<td>float</td>
<td>0.0 to 1.0</td>
</tr>
</tbody>
</table>

Guides object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>string</td>
<td>A color string that specifies the color used for the guides (see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>hGuides</td>
<td>array</td>
<td>Array of floating-point numbers that specify horizontal guide locations.</td>
</tr>
<tr>
<td>locked</td>
<td>Boolean</td>
<td>If set to true, the user cannot select or move the guides. The default value is false.</td>
</tr>
<tr>
<td>vGuides</td>
<td>array</td>
<td>Array of floating-point numbers that specify vertical guide locations.</td>
</tr>
</tbody>
</table>
Layer object

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 (•).

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>disclosure</td>
<td>Boolean</td>
<td>If set to true, the Layers list displays all the objects in the layer. If set to false, only the name of the layer appears.</td>
</tr>
<tr>
<td>elems •</td>
<td>array</td>
<td>Array of elements inside a layer which also include sublayers.</td>
</tr>
<tr>
<td>frames •</td>
<td>array</td>
<td>An array of FrameNLayerIntersection objects (see “FrameNLayerIntersection object” on page 266).</td>
</tr>
<tr>
<td>isLayer •</td>
<td>Boolean</td>
<td>Always true for a layer.</td>
</tr>
<tr>
<td>layerType •</td>
<td>string</td>
<td>Acceptable values are ”normal” and ”web”.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Might be null (removes any existing name).</td>
</tr>
<tr>
<td>sharing</td>
<td>string</td>
<td>Acceptable values are ”shared” and ”not shared”.</td>
</tr>
</tbody>
</table>

PathAttrs object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>brush</td>
<td>object</td>
<td>Brush object (see ”Brush object” on page 243).</td>
</tr>
<tr>
<td>brushColor</td>
<td>string</td>
<td>A color string that specifies the color that is used for rendering the Brush object, if any (see “Color string data type” on page 8).</td>
</tr>
<tr>
<td>brushPlacement</td>
<td>string</td>
<td>Acceptable values are ”inside”, ”center”, and ”outside”.</td>
</tr>
<tr>
<td>brushTexture</td>
<td>object</td>
<td>Texture object (see ”Texture object” on page 259).</td>
</tr>
<tr>
<td>fill</td>
<td>object</td>
<td>Fill object (see ”Fill object” on page 265).</td>
</tr>
<tr>
<td>fillColor</td>
<td>string</td>
<td>A color string that specifies the color used for rendering the Fill object, if any (see ”Color string data type” on page 8).</td>
</tr>
<tr>
<td>fillHandle1</td>
<td>point</td>
<td>The three fillHandle properties are used by Gradient and Pattern fills to set the angle and size of the gradient/pattern.</td>
</tr>
<tr>
<td>fillHandle2</td>
<td>point</td>
<td></td>
</tr>
<tr>
<td>fillHandle3</td>
<td>point</td>
<td></td>
</tr>
<tr>
<td>fillOnTop</td>
<td>Boolean</td>
<td>If set to true, the fill is drawn on top of the brush; if set to false (the default), the fill is drawn beneath the brush.</td>
</tr>
<tr>
<td>fillTexture</td>
<td>object</td>
<td>Texture object (see ”Texture object” on page 259).</td>
</tr>
</tbody>
</table>
Pattern object

The following table lists the property of the Pattern object, along with its data type and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>The name that appears in the Fill panel.</td>
</tr>
</tbody>
</table>

RectanglePrimitive object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>roundness</td>
<td>float</td>
<td>A floating-point value between 0 and 1 that specifies the &quot;roundness&quot; to use for the corners (0 is no roundness, 1 is 100% roundness).</td>
</tr>
<tr>
<td>originalSides</td>
<td>rectangle</td>
<td>A rectangle that specifies the original sides of the primitive (see &quot;Rectangle data type&quot; on page 9). Because rectangle primitives remember transformations, the user might see something different from the original sides.</td>
</tr>
<tr>
<td>mode</td>
<td>string</td>
<td>Specifies the roundness units, which can be relative (percentage) or exact (pixels).</td>
</tr>
<tr>
<td>transform</td>
<td>matrix</td>
<td>A matrix that indicates all the transformations that were applied to the primitive (see &quot;Matrix data type&quot; on page 9).</td>
</tr>
<tr>
<td>pathAttributes</td>
<td>object</td>
<td>A PathAttrs object that indicates the path attributes of the primitive (see &quot;PathAttrs object&quot; on page 268).</td>
</tr>
</tbody>
</table>

RegisterMoveParms object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>constrainAngles</td>
<td>Boolean</td>
<td>Determines whether dragging the control point constrains the angles to the minAngle and maxAngle values.</td>
</tr>
<tr>
<td>constrainRotateKey</td>
<td>string</td>
<td>Pass in the key that you want to use to constrain the rotation. A value of &quot;none&quot; means that rotation will not be constrained. A value of &quot;shiftKey&quot; means that when the user holds down the Shift key while dragging the mouse, rotation will be constrained. The value can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;. Note: these points are set with minAngle and maxAngle.</td>
</tr>
<tr>
<td>constrainX</td>
<td>float</td>
<td>The value to constrain the x coordinate. Note: the method constrainXKey must be used with this method.</td>
</tr>
<tr>
<td>Property</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>constrainXKey</td>
<td>string</td>
<td>Pass in the key that you want to use to constrain the x-coordinate value. A value of &quot;none&quot; means that x will not be constrained. A value of &quot;shiftKey&quot; means that when the user holds down the Shift key while dragging the mouse, x will be constrained. The value can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;.</td>
</tr>
<tr>
<td>constrainY</td>
<td></td>
<td>The value to constrain the y coordinate.</td>
</tr>
<tr>
<td>constrainYKey</td>
<td></td>
<td>Pass in the key that you want to use to constrain the y-coordinate value. A value of &quot;none&quot; means that y will not be constrained. A value of &quot;shiftKey&quot; means that when the user holds down the Shift key while dragging the mouse, the value of y will be constrained. The value can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;.</td>
</tr>
<tr>
<td>constrain45Key</td>
<td>string</td>
<td>The key value that you want to use to constrain movement to the nearest 45º increment. Can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;. A key value of &quot;none&quot; means dragging will not be constrained, &quot;shiftKey&quot; (or other value) means that when the user holds down the Shift key (or other value) while dragging, movement will be constrained.</td>
</tr>
<tr>
<td>constrain90Key</td>
<td>string</td>
<td>The key value that you want to use to constrain movement to the nearest 90º increment. Can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;. A key value of &quot;none&quot; means dragging will not be constrained, &quot;shiftKey&quot; (or other key) means that when the user holds down the Shift key (or other key) while dragging, movement will be constrained.</td>
</tr>
<tr>
<td>deltaLinearToLinear</td>
<td>float</td>
<td>Determines the ratio of mouse movement to point movement along the line. For example, a value of 1.0 means that if the mouse moves 1 pixel, the point moves 1 pixel along the line specified in the method RegisterLinearMove.</td>
</tr>
<tr>
<td>deltaRtoR</td>
<td>float</td>
<td>Determines the mouse radius change relative to the point radius change. For example, a value of 1.0 means that as the mouse moves 1 pixel away from the center of the object, the point also moves 1 pixel away from the center of the object.</td>
</tr>
<tr>
<td>deltaShortestSideToX</td>
<td>float</td>
<td>The ratio of shortest mouse movement to the movement of referenced point’s x coordinate.</td>
</tr>
<tr>
<td>deltaShortestSideToY</td>
<td>float</td>
<td>The ratio of shortest mouse movement to the movement of referenced point’s y coordinate.</td>
</tr>
<tr>
<td>deltaLongestSideToX</td>
<td>float</td>
<td>The ratio of longest mouse movement to the movement of referenced point’s x coordinate.</td>
</tr>
<tr>
<td>deltaLongestSideToY</td>
<td>float</td>
<td>The ratio of longest mouse movement to the movement of referenced point’s y coordinate.</td>
</tr>
<tr>
<td>deltaXtoX</td>
<td>float</td>
<td>The ratio of mouse movement to the movement of the referenced point’s x coordinate. For example, 1.0 means that when the mouse moves 1 pixel to the right, the referenced point also moves 1 pixel to the right.</td>
</tr>
<tr>
<td>Property</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>deltaXtoY</td>
<td>float</td>
<td>The ratio of mouse movement on the x-axis to the movement of the referenced point's y coordinate. For example, 1.0 means that when the mouse moves 1 pixel to the left, the referenced point moves 1 pixel towards the top of the document.</td>
</tr>
<tr>
<td>deltaYtoX</td>
<td>float</td>
<td>The ratio of mouse movement on the y-axis to the movement of the referenced point's x coordinate. For example, 1.0 means that when the mouse moves 1 pixel to the top of the document, the referenced point moves 1 pixel to the left.</td>
</tr>
<tr>
<td>deltaYtoY</td>
<td>float</td>
<td>The ratio of mouse movement to the movement of the referenced point's y coordinate. For example, 1.0 means that when the mouse moves 1 pixel toward the bottom of the document, the referenced point also moves 1 pixel toward the bottom of the document.</td>
</tr>
<tr>
<td>disableRotateKey</td>
<td>string</td>
<td>Pass in the key that you want to use to disable rotating around the center. The value can be one of the following: &quot;none&quot;, &quot;shiftKey&quot;, &quot;ctrlCmdKey&quot;, &quot;altOptKey&quot;. A value of &quot;none&quot; means rotation will not be constrained. A value of &quot;shiftKey&quot; means that when the user holds down the Shift key while dragging the mouse, rotation is not constrained.</td>
</tr>
<tr>
<td>incrementRadius</td>
<td>float</td>
<td>Constant value that is added to the radius.</td>
</tr>
<tr>
<td>incrementX</td>
<td>float</td>
<td>This amount is added to the x movement of the mouse when calculating the total movement.</td>
</tr>
<tr>
<td>incrementY</td>
<td>float</td>
<td>This amount is added to the y movement of the mouse when calculating the total movement.</td>
</tr>
<tr>
<td>maxAngle</td>
<td>point</td>
<td>The maximum angle that can be set.</td>
</tr>
<tr>
<td>maxLinear</td>
<td>float</td>
<td>Determines the maximum amount the point can move along a line.</td>
</tr>
<tr>
<td>maxRadius</td>
<td>float</td>
<td>The maximum radius value.</td>
</tr>
<tr>
<td>maxX</td>
<td>float</td>
<td>The maximum value the x coordinate can move.</td>
</tr>
<tr>
<td>maxY</td>
<td>float</td>
<td>The maximum value the y coordinate can move.</td>
</tr>
<tr>
<td>minAngle</td>
<td>point</td>
<td>The minimum angle that can be set.</td>
</tr>
<tr>
<td>minLinear</td>
<td>float</td>
<td>The minimum amount the point can move along a line.</td>
</tr>
<tr>
<td>minMaxRelative</td>
<td>Boolean</td>
<td>Determines whether the min and max values are relative or absolute. For example, if max.x=100 and minMaxRelative is true, then x can move up 100 points to the right. If minMaxRelative is set to false then the maximum x can be set to is 100.</td>
</tr>
<tr>
<td>minX</td>
<td>float</td>
<td>The minimum value the x coordinate can move.</td>
</tr>
<tr>
<td>minY</td>
<td>float</td>
<td>The minimum value the y coordinate can move.</td>
</tr>
<tr>
<td>minRadius</td>
<td>float</td>
<td>The minimum radius value.</td>
</tr>
<tr>
<td>movePred</td>
<td>Boolean</td>
<td>Determines whether the predecessor point should be moved as the user moves the mouse.</td>
</tr>
</tbody>
</table>
SingleTextRun object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>changedAttrs</td>
<td>object</td>
<td>TextAttrs object (see “TextAttrs object” on page 274).</td>
</tr>
<tr>
<td>characters</td>
<td>string</td>
<td>The text that is contained in this run.</td>
</tr>
</tbody>
</table>

SmartShape object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altOptKeyDown •</td>
<td>Boolean</td>
<td>Indicates whether the Alt/Option key is pressed (true if pressed, otherwise false).</td>
</tr>
<tr>
<td>constrainDragInsertAspect</td>
<td>Boolean</td>
<td>Determines if, while dragging a shape on the canvas, the aspect ratio is constrained (true if constrained, otherwise false).</td>
</tr>
<tr>
<td>constrainDragInsertAspectKey</td>
<td>string</td>
<td>The key value that will cause the aspect ratio to be constrained during a DragInsert operation.</td>
</tr>
<tr>
<td>ctrlCmdKeyDown •</td>
<td>Boolean</td>
<td>Indicates whether the Control/Command key is pressed (true if pressed, otherwise false).</td>
</tr>
<tr>
<td>currentControlPoint •</td>
<td>object</td>
<td>Returns the current control point object.</td>
</tr>
<tr>
<td>currentControlPointIndex •</td>
<td>integer</td>
<td>Returns the index number of the current control point.</td>
</tr>
<tr>
<td>currentControlPointName •</td>
<td>string</td>
<td>Returns the name of the current control point.</td>
</tr>
<tr>
<td>currentMousePos</td>
<td>point</td>
<td>Location of the mouse during the current drag message.</td>
</tr>
<tr>
<td>elem •</td>
<td>object</td>
<td>Objects defined as part of the current Auto Shape.</td>
</tr>
<tr>
<td>getsDragEvents</td>
<td>Boolean</td>
<td>Sets notification for drag events (true notifies the smartshape for every mouse movement, false sets no notification).</td>
</tr>
</tbody>
</table>
The following table lists the method of the SmartShape object, along with its parameter.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDefaultMoveParms()</td>
<td>object</td>
<td>Returns an object that has all of the default move parameters set.</td>
</tr>
</tbody>
</table>

**Style object**

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

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectList</td>
<td>object</td>
<td>EffectList object (see “EffectList object” on page 254).</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The name displayed in the Style panel.</td>
</tr>
<tr>
<td>pathAttributes</td>
<td>object</td>
<td>PathAttrs object (see “PathAttrs object” on page 268).</td>
</tr>
<tr>
<td>tdTagText</td>
<td>string</td>
<td>A string that contains all the attributes of a table cell except colspan and rowspan. Should be in a format similar to the following: &quot;bgcolor=&quot;ff0000&quot; valign=&quot;top&quot;&quot;</td>
</tr>
<tr>
<td>textBold</td>
<td>Boolean</td>
<td>Makes the specified text bold; used only if use_textStyles is set to true.</td>
</tr>
<tr>
<td>textFont</td>
<td>string</td>
<td>The font to apply to text; used only if use_textFont is set to true.</td>
</tr>
<tr>
<td>textItalic</td>
<td>Boolean</td>
<td>Makes the specified text italic; used only if use_textStyles is set to true.</td>
</tr>
<tr>
<td>textSize</td>
<td>string</td>
<td>String of the form &quot;#pt&quot;, where # is a numeric value.</td>
</tr>
<tr>
<td>textUnderline</td>
<td>Boolean</td>
<td>Underlines the specified text; used only if use_textStyles is set to true.</td>
</tr>
<tr>
<td>use_brush</td>
<td>Boolean</td>
<td>If set to true, applies the brush property of the PathAttrs object when applying the style. If set to false, ignores the brush property. The default value is false.</td>
</tr>
</tbody>
</table>
The following table lists the properties of the TextAttrs object, along with their data types and, where appropriate, acceptable values and notes.

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_brushColor</td>
<td>Boolean</td>
<td>If set to true, applies the brushColor property of the PathAttrs object when applying the style. If set to false, ignores the brushColor property. The default value is false.</td>
</tr>
<tr>
<td>use_effectList</td>
<td>Boolean</td>
<td>If set to true, applies the effects property of the EffectList object when applying the style. If set to false, ignores the effects property. The default value is false.</td>
</tr>
<tr>
<td>use_fill</td>
<td>Boolean</td>
<td>If set to true, applies the fill property of the PathAttrs object when applying the style. If set to false, ignores the fill property. The default value is false.</td>
</tr>
<tr>
<td>use_fillColor</td>
<td>Boolean</td>
<td>If set to true, applies the fillColor property of the PathAttrs object when applying the style. If set to false, ignores the fillColor property. The default value is false.</td>
</tr>
<tr>
<td>use_textFont</td>
<td>Boolean</td>
<td>If set to true, applies the textFont property of the PathAttrs object when applying the style. If set to false, ignores the textFont property. The default value is false.</td>
</tr>
<tr>
<td>use_textSize</td>
<td>Boolean</td>
<td>If set to true, applies the textSize property of the PathAttrs object when applying the style. If set to false, ignores the textSize property. The default value is false.</td>
</tr>
<tr>
<td>use_textStyles</td>
<td>Boolean</td>
<td>If set to true, applies the textStyles property of the PathAttrs object when applying the style. If set to false, ignores the textStyles property. The default value is false.</td>
</tr>
</tbody>
</table>

**TextAttrs object**

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignment</td>
<td>string</td>
<td>Acceptable values are &quot;left&quot;, &quot;center&quot;, &quot;right&quot;, &quot;justify&quot;, and &quot;stretch&quot;.</td>
</tr>
<tr>
<td>baselineShift</td>
<td>integer</td>
<td>The number of pixels above (positive numbers) or below (negative numbers) the baseline by which the characters are shifted.</td>
</tr>
<tr>
<td>bold</td>
<td>Boolean</td>
<td>Set to true for bold text, false for normal text.</td>
</tr>
<tr>
<td>face</td>
<td>string</td>
<td>The name of the font, such as Arial.</td>
</tr>
<tr>
<td>fillColor</td>
<td>string</td>
<td>A color string that specifies the color of the text (see &quot;Color string data type&quot; on page 8).</td>
</tr>
</tbody>
</table>
| horizontalScale| float    | The relative width of the characters.  
1.0 — normal width  
< 1 — thinner than normal  
> 1 — wider than normal |
| italic       | Boolean   | Set to true for italic text, false for normal text.               |
TextRuns object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialAttrs</td>
<td>object</td>
<td>TextAttrs object (see &quot;TextAttrs object&quot; on page 274).</td>
</tr>
<tr>
<td>textRuns</td>
<td>array</td>
<td>Array of SingleTextRun objects on this TextRuns object (see &quot;SingleTextRun object&quot; on page 272).</td>
</tr>
</tbody>
</table>

Widget object

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>element.visible</td>
<td>Boolean</td>
<td>Sets the visibility of an element to true or false.</td>
</tr>
<tr>
<td>element.opacity</td>
<td>integer</td>
<td>Sets the opacity of an element.</td>
</tr>
<tr>
<td>element.pathattrs.brushColor</td>
<td>string</td>
<td>A color string that specifies the brush color for the path attributes of the primitive (see &quot;Color string data type&quot; on page 8 and &quot;PathAttrs object&quot; on page 268).</td>
</tr>
</tbody>
</table>
The following table lists a method of the Widget object, along with its parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetObjectByName()</td>
<td>elemName</td>
<td>The elemName value specifies the element name for the object as it is specified in the Layers panel. This method returns the JavaScript type for the specified element name. For example: <code>var bound_rect = Widget.GetObjectByName(elemName);</code></td>
</tr>
<tr>
<td>isWidgetSelected</td>
<td>Boolean</td>
<td>true if the selected element is a widget.</td>
</tr>
<tr>
<td>UpdateWidget</td>
<td>propString</td>
<td>Updates the rich symbol instances.</td>
</tr>
<tr>
<td>Widget.elem</td>
<td>elemName</td>
<td>Reference to the instance currently executing the widget script.</td>
</tr>
<tr>
<td>Widget.opCode</td>
<td>String</td>
<td>A num code representing the action for the current running widget script. 1 = setDefaultValues, 2 = applyCurrentValues.</td>
</tr>
<tr>
<td>widget.jsCode</td>
<td>String</td>
<td>The jsCode for the currently running widget script.</td>
</tr>
</tbody>
</table>
Chapter 6: 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, see 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 8.

BehaviorInfo object

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 object properties are read-only.

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>integer</td>
<td>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;</td>
</tr>
<tr>
<td>behaviorText</td>
<td>string</td>
<td>For roundtrip HTML from Dreamweaver, specifies the JavaScript behaviors that don’t have an equivalent in Fireworks.</td>
</tr>
<tr>
<td>borderColor</td>
<td>string</td>
<td>If action is set to 9 (Popup Menu), specifies the border color of the menu, in hexadecimal.</td>
</tr>
<tr>
<td>borderSize</td>
<td>integer</td>
<td>If action is set to 9 (Popup Menu), specifies the size of the menu border in points.</td>
</tr>
<tr>
<td>cellOverColor</td>
<td>string</td>
<td>If action is set to 9 (Popup Menu), specifies the cell color for the Over state.</td>
</tr>
<tr>
<td>cellUpColor</td>
<td>string</td>
<td>If action is set to 9 (Popup Menu), specifies the cell color for the Up state.</td>
</tr>
<tr>
<td>creationDate</td>
<td>date</td>
<td>Specifies the date the document was created.</td>
</tr>
<tr>
<td>dhHref</td>
<td>string</td>
<td>If action is set to 6 (Button Highlight), specifies the URL of the highlight image used for the Down button state.</td>
</tr>
</tbody>
</table>
## Property (read-only) Data type Notes

<table>
<thead>
<tr>
<th>Property</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhTargetFrame</td>
<td>integer</td>
<td>If <code>action</code> is set to 6 (Button Highlight), specifies the target frame number for the down highlight state.</td>
</tr>
<tr>
<td>downHighlight</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 6 (Button Highlight), specifies if there is an image highlight for the Down button state.</td>
</tr>
<tr>
<td>event</td>
<td>integer</td>
<td>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: <code>var kEventMouseOver = 0; var kEventOnClick = 1; var kEventMouseOut = 2; var kEventOnLoad = 3;</code></td>
</tr>
<tr>
<td>groupChecked</td>
<td>Boolean</td>
<td>Specifies whether the group option is checked.</td>
</tr>
<tr>
<td>groupName</td>
<td>string</td>
<td>Returns a constant value such as navbar1.</td>
</tr>
<tr>
<td>hasDhTargetFrame</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 6 (Button Highlight), specifies if the highlight image for the Down button state has a target frame.</td>
</tr>
<tr>
<td>hasDhHref</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 6 (Button Highlight), specifies if the highlight image for the Down button state has an href.</td>
</tr>
<tr>
<td>hasHref</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies if an external file is swapped in. The value of hasHref is always the opposite of hasTargetFrame; you cannot swap from two sources.</td>
</tr>
<tr>
<td>hasStatusText</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 1 (Status Message), specifies if the status text is not empty.</td>
</tr>
<tr>
<td>hasTargetFrame</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies if the swap image swaps is in another frame in the Fireworks file. The value of hasTargetFrame is always the opposite of hasHref; you cannot swap from two sources.</td>
</tr>
<tr>
<td>hideTimeout</td>
<td>Integer</td>
<td>Specifies the timeout in milliseconds.</td>
</tr>
<tr>
<td>hideOnMouseout</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies if the menu is hidden on a Mouse Out event.</td>
</tr>
<tr>
<td>hiliteColor</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the highlight color in the upper-left corner of the menu cells.</td>
</tr>
<tr>
<td>horzOffset</td>
<td>integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), <code>horzOffset</code> specifies the horizontal pixel offset for the menu.</td>
</tr>
<tr>
<td>horzSubmenuOffset</td>
<td>Integer</td>
<td>Specifies the horizontal offset value of the submenu.</td>
</tr>
<tr>
<td>href</td>
<td>string</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies the file URL for an external swap image file. Value is expressed as <code>file://URL</code>.</td>
</tr>
<tr>
<td>menuFontFamily</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the name of the font family to use for the menu.</td>
</tr>
<tr>
<td>menuHeight</td>
<td>Integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the height in points of the menu cell.</td>
</tr>
<tr>
<td>menuImagePath</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the path to the first image.</td>
</tr>
<tr>
<td>menuImagePath2</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the path to the second image.</td>
</tr>
<tr>
<td>menuItems</td>
<td>array</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies an array that lists the items in the menu.</td>
</tr>
<tr>
<td>Property (read-only)</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>menuItemPadding</td>
<td>integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the cell padding for the menu items.</td>
</tr>
<tr>
<td>menuItemSpacing</td>
<td>integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the spacing between menu items in points.</td>
</tr>
<tr>
<td>menuWidth</td>
<td>integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the width, in points, of the menu cell.</td>
</tr>
<tr>
<td>objectName</td>
<td>string</td>
<td>Name of the behavior.</td>
</tr>
<tr>
<td>opaqueBackground</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies if the menu cell background is opaque.</td>
</tr>
<tr>
<td>overrideMenuHeight</td>
<td>Boolean</td>
<td>Specifies whether the menu height is overridden.</td>
</tr>
<tr>
<td>overrideMenuWidth</td>
<td>Boolean</td>
<td>Specifies whether the menu width is overridden.</td>
</tr>
<tr>
<td>preload</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies if the image is to be preloaded.</td>
</tr>
<tr>
<td>restoreOnMouseout</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies if the original image is restored on a Mouse Out event.</td>
</tr>
<tr>
<td>shadowColor</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the shadow color in the lower-right corner of the menu cells.</td>
</tr>
<tr>
<td>soundFileName</td>
<td>string</td>
<td>URL of the sound file.</td>
</tr>
<tr>
<td>statusText</td>
<td>string</td>
<td>If <code>action</code> is set to 1 (Status Message), specifies the status message text.</td>
</tr>
<tr>
<td>submenuRelativeToItem</td>
<td>Boolean</td>
<td>Specifies whether the submenu is relative to the menu item.</td>
</tr>
<tr>
<td>targetColumnNum</td>
<td>zero-based index</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies the column in the slices table that is swapped.</td>
</tr>
<tr>
<td>targetFrameNum</td>
<td>zero-based index</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies the frame number to be swapped if <code>hasTargetFrame</code> is set to <code>true</code>.</td>
</tr>
<tr>
<td>targetRowNum</td>
<td>zero-based index</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies the row in the slices table that is swapped.</td>
</tr>
<tr>
<td>targetTable</td>
<td>object</td>
<td>If <code>action</code> is set to 2 (Swap Image), specifies the table of slices in the target swap frame.</td>
</tr>
<tr>
<td>textAlignment</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the alignment for the menu text. Acceptable values are &quot;left&quot;, &quot;center&quot;, and &quot;right&quot;.</td>
</tr>
<tr>
<td>textBold</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 9 (Popup Menu), true if the menu text is bold.</td>
</tr>
<tr>
<td>textFamily</td>
<td>string</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the font family to use for the menu text.</td>
</tr>
<tr>
<td>textIndent</td>
<td>integer</td>
<td>If <code>action</code> is set to 9 (Popup Menu), specifies the left indent, in points, of the menu text.</td>
</tr>
<tr>
<td>textItalic</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 9 (Popup Menu), true if the menu text is italic.</td>
</tr>
<tr>
<td>textOnly</td>
<td>Boolean</td>
<td>If <code>action</code> is set to 9 (Popup Menu), true if the pop-up menu is to be text only.</td>
</tr>
</tbody>
</table>
The BehaviorsList object is an array of BehaviorInfo objects that describe the behaviors in an image map (see “BehaviorInfo object” on page 277). 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.

```javascript
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.

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>numberOfBehaviors</td>
<td>integer</td>
<td>The number of BehaviorInfo objects in the BehaviorsList array (0 or more) (see “BehaviorInfo object” on page 277).</td>
</tr>
</tbody>
</table>

**exportDoc object**

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.*

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altText</td>
<td>string</td>
<td>The alternate text description for the Fireworks document.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>string</td>
<td>The hexadecimal color of the document canvas, without the # character; for example, &quot;#FF0000&quot; for red background.</td>
</tr>
<tr>
<td>backgroundAttachment</td>
<td>string</td>
<td>Specifies whether the document has a background image. If an image is set in the HTML setup, returns true.</td>
</tr>
<tr>
<td>backgroundIsTransparent</td>
<td>Boolean</td>
<td>Set to true if the Fireworks canvas color is transparent or if the export settings specify a transparent GIF format; false otherwise.</td>
</tr>
<tr>
<td>backgroundLink</td>
<td>string</td>
<td>The background URL, which is expressed as file:///URL.</td>
</tr>
</tbody>
</table>
### HTML export objects

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>backgroundRepeat</td>
<td>string</td>
<td>Specifies how the background image repeats as X, Y, Repeat (Both) and No Repeat.</td>
</tr>
<tr>
<td>backgroundURL</td>
<td>string</td>
<td>The background image URL, which is expressed as file:///URL.</td>
</tr>
<tr>
<td>bottomMargin</td>
<td>long</td>
<td>Specifies the bottom margin of the page.</td>
</tr>
<tr>
<td>cssPopupMenus</td>
<td>Boolean</td>
<td>If this value is true, Fireworks outputs combined CSS/JavaScript pop-up menus; if false, Fireworks outputs JavaScript-only pop-up menus.</td>
</tr>
<tr>
<td>docAlignment</td>
<td>string</td>
<td>Specifies the page alignment as left, center, right. The default is left.</td>
</tr>
<tr>
<td>docID</td>
<td>integer</td>
<td>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 &gt; Save As, you can get multiple files with the same docID.</td>
</tr>
<tr>
<td>docSaveFolder</td>
<td>string</td>
<td>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.</td>
</tr>
<tr>
<td>docSaveName</td>
<td>string</td>
<td>The filename used when the document was saved, without path information, such as &quot;nav.gif&quot;.</td>
</tr>
<tr>
<td>documentXHTMLFormat</td>
<td>Boolean</td>
<td>Determines whether Fireworks outputs XHTML formatted files. DO NOT USE - Use &quot;xhtmlFormat&quot; instead. Returns true if the file is XHTML formatted files; false if the file is HTML formatted when the user exports a file.</td>
</tr>
<tr>
<td>emptyCellColor</td>
<td>string</td>
<td>A color string that specifies the color of empty table cells (see &quot;Color string data type&quot; on page 8).</td>
</tr>
<tr>
<td>emptyCellContents</td>
<td>integer</td>
<td>Specifies what to put into empty cells. Acceptable values are 1 (nothing), 2 (spacer image), and 3 (nonbreaking space).</td>
</tr>
<tr>
<td>emptyCellUsesCanvasColor</td>
<td>Boolean</td>
<td>If set to true (the default), empty cells are set to the backgroundColor property. If set to false, they are set to the emptyCellColor property.</td>
</tr>
<tr>
<td>externalCSS</td>
<td>Boolean</td>
<td>If set to true, Fireworks outputs an external CSS file.</td>
</tr>
<tr>
<td>externalCSSFileName</td>
<td>string</td>
<td>The name of the external CSS file.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>URL for the exported image, relative to the HTML output; for example, &quot;images/Button.gif&quot;. 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 &quot;Button_r2_c2.gif&quot;.</td>
</tr>
<tr>
<td>generateHeader</td>
<td>Boolean</td>
<td>Set to true if an HTML file is generated; false if the output goes to the clipboard.</td>
</tr>
<tr>
<td>hasAltText</td>
<td>Boolean</td>
<td>Set to true if the Fireworks document has an alternate text description.</td>
</tr>
<tr>
<td>hasBackgroundColor</td>
<td>Boolean</td>
<td>Set to true if the Fireworks document has a background URL.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the image that is being exported, in pixels. In the Slices.htt file, it is the total height of the output images.</td>
</tr>
<tr>
<td>htmlEncoding</td>
<td>string</td>
<td>Determines the encoding standard for the HTML file that Fireworks generates during export. Use &quot;iso-8859-1&quot; for ASCII or &quot;utf-8&quot; for Unicode.</td>
</tr>
</tbody>
</table>
### Property (read-only) | Data type | Notes
--- | --- | ---
htmlOutputPath | string | File that the HTML is being written to, including the filename, which is expressed as file://URL; for example, "file:///C:/top/nav/navbar.htm".
imagename | string | Name of the image that is being exported, without the 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.
isBackground | Boolean | Specifies whether the document has a background image. If an image is set in the HTML set up as the background image, return true.
leftMargin | long | Specifies the left margin of the page.
umFrames | integer | Number of frames that are being exported from the Fireworks document. This value is not zero-based; the value is 1 or more.
pagename | string | Name of the page.
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".
rightMargin | long | Specifies the right margin of the page.
serverMap | Boolean | Specifies whether the document uses a server map.
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".
symbols | array | List of symbols in the document.
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"".
tablePctWidth | string | A string that contains the percentage width of the table. If it returns 0, there is no percentage width and the pixel width of the table should be used. 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"align="center"". If the table is right-aligned, the string is "align="right"align="right"".
topMargin | long | Specifies the top margin of the page.
width | integer | Width of the image being exported, in pixels. In the Slices.htm file, it is the total width of the output images.
xhtmlFormat | Boolean | Determines whether Fireworks outputs XHTML-formatted files (true) or HTML-formatted files (false) when the user exports a file.
ImageMap object

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 object properties are read-only.

<table>
<thead>
<tr>
<th>Property (read-only) or Method</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altText</td>
<td>string</td>
<td>The alternate text description for this slice, if any.</td>
</tr>
<tr>
<td>behaviors</td>
<td>object</td>
<td>BehaviorsList object that contains the behaviors for this slice (see &quot;BehaviorsList object&quot; on page 280).</td>
</tr>
<tr>
<td>hasAltText</td>
<td>Boolean</td>
<td>Set to true if the slice has an alternate text description.</td>
</tr>
<tr>
<td>hasHref</td>
<td>Boolean</td>
<td>Set to true if the slice has a URL.</td>
</tr>
<tr>
<td>hasTargetText</td>
<td>Boolean</td>
<td>Set to true if the target text is not empty.</td>
</tr>
<tr>
<td>href</td>
<td>string</td>
<td>The URL link for this slice, which is expressed as file://URL.</td>
</tr>
<tr>
<td>left</td>
<td>string</td>
<td>Left coordinates of the image map.</td>
</tr>
<tr>
<td>numCoords</td>
<td>integer</td>
<td>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.</td>
</tr>
<tr>
<td>radius</td>
<td>integer</td>
<td>Radius of the area, if shape is &quot;circle&quot;.</td>
</tr>
<tr>
<td>shape</td>
<td>string</td>
<td>Acceptable values are &quot;circle&quot;, &quot;poly&quot;, and &quot;rect&quot;.</td>
</tr>
<tr>
<td>targetText</td>
<td>string</td>
<td>Target text for this image, if any.</td>
</tr>
<tr>
<td>top</td>
<td>string</td>
<td>Top coordinates of the image map.</td>
</tr>
</tbody>
</table>
| xCoord(index)                 | zero-based index | 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 index | Returns the y coordinate for the specified point, in pixels. See xCoord(). |

ImagemapList object

The ImagemapList object is an array of ImageMap objects that describe the areas in an image map (see "ImageMap object" on page 283). To access ImageMap objects, use the ImagemapList array, as shown below:

```javascript
var curImagemap = ImagemapList[1];
```

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

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>numberOfURLs</td>
<td>integer</td>
<td>The number of image map areas in the image map list (0 or more).</td>
</tr>
</tbody>
</table>
### SliceInfo object

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 object properties are read-only.

<table>
<thead>
<tr>
<th>Property (read-only) or method</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>altText</td>
<td>string</td>
<td>The alternate text description for this slice.</td>
</tr>
<tr>
<td>backgroundAttachment</td>
<td>string</td>
<td>Specifies if the background image is fixed or scrolls with the rest of the page.</td>
</tr>
<tr>
<td>backgroundHorizontalPosition</td>
<td>string</td>
<td>Horizontal position of the background image. (Left, right, or center, numeric value)</td>
</tr>
<tr>
<td>backgroundHorizontalPositionVal</td>
<td>long</td>
<td>Horizontal position value of the background image if the argument is value.</td>
</tr>
<tr>
<td>backgroundRepeat</td>
<td>string</td>
<td>Specifies if the background image repeats, and if repeats in X, Y, Repeat (Both) and No Repeat.</td>
</tr>
<tr>
<td>backgroundVerticalPosition</td>
<td>string</td>
<td>Vertical position of the background image. (Left, right, or center, numeric value).</td>
</tr>
<tr>
<td>backgroundVerticalPositionVal</td>
<td>long</td>
<td>Vertical position value of the background image if the argument is value.</td>
</tr>
<tr>
<td>behaviors</td>
<td>object</td>
<td>BehaviorsList object that contains the behaviors for this slice (see &quot;BehaviorsList object&quot; on page 280).</td>
</tr>
<tr>
<td>cellHeight</td>
<td>integer</td>
<td>Height of this table row, in pixels.</td>
</tr>
<tr>
<td>cellWidth</td>
<td>integer</td>
<td>Width of this table column, in pixels.</td>
</tr>
<tr>
<td>columnSpan</td>
<td>integer</td>
<td>Span of the column.</td>
</tr>
<tr>
<td>downIndex</td>
<td>zero-based index</td>
<td>The index for the frame of the down state for button slices.</td>
</tr>
<tr>
<td>getFrameFileName (frameIndex)</td>
<td>zero-based index</td>
<td>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 &quot;Button_r1_c1&quot;. 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.</td>
</tr>
<tr>
<td>hasAltText</td>
<td>Boolean</td>
<td>Set to true if the slice has an alternate text description.</td>
</tr>
<tr>
<td>hasHref</td>
<td>Boolean</td>
<td>Set to true if the slice has a URL.</td>
</tr>
<tr>
<td>hasHtmlText</td>
<td>Boolean</td>
<td>Set to true if the cell is a text-only slice.</td>
</tr>
<tr>
<td>hasImage</td>
<td>Boolean</td>
<td>Set to true if this cell has an image. For text-only slices, this is set to false.</td>
</tr>
<tr>
<td>hasImagemap</td>
<td>Boolean</td>
<td>Set to true if there are image map Hotspots in this image slice.</td>
</tr>
<tr>
<td>hasTargetText</td>
<td>Boolean</td>
<td>Set to true if the target text is not empty.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the image in pixels, including row spans.</td>
</tr>
<tr>
<td>Property (read-only) or method</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>href</td>
<td>string</td>
<td>The URL link for this slice, which is expressed as file://URL.</td>
</tr>
<tr>
<td>htmlText</td>
<td>string</td>
<td>Text for a text-only slice.</td>
</tr>
<tr>
<td>imagemap</td>
<td>object</td>
<td>ImagemapList object containing the image map information for this slice (see “ImagemapList object” on page 283).</td>
</tr>
<tr>
<td>imageSuffix</td>
<td>string</td>
<td>Extension for the image in this cell, including a period (); for example, &quot;.gif&quot;.</td>
</tr>
<tr>
<td>isBackground</td>
<td>Boolean</td>
<td>Specifies whether the slice is background image or foreground image. True = Background.</td>
</tr>
<tr>
<td>isUndefined</td>
<td>Boolean</td>
<td>Set to 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.</td>
</tr>
<tr>
<td>left</td>
<td>integer</td>
<td>Left side of the cell in pixels. The left starts at 0.</td>
</tr>
<tr>
<td>nestedTableSlices</td>
<td>object</td>
<td>A Slices object that describes a nested table occupying the current table cell (see &quot;Slices object&quot; on page 286). Set to null if the cell does not contain a nested table.</td>
</tr>
<tr>
<td>rowSpan</td>
<td>integer</td>
<td>Span of the row.</td>
</tr>
<tr>
<td>setFrameFileName(frameIndex)</td>
<td>zero-based index</td>
<td>Sets the filename for the slice on the specified frame, without directory or extension information. You can stop an image from being exported by setting its name to &quot;&quot; (an empty string).</td>
</tr>
<tr>
<td>skipCell</td>
<td>Boolean</td>
<td>Set to true if this cell in the table is covered by a previous row span or column span.</td>
</tr>
<tr>
<td>tableAlign</td>
<td>string</td>
<td>The table alignment for the table in the current cell.</td>
</tr>
<tr>
<td>tableBorder</td>
<td>integer</td>
<td>The table’s border width.</td>
</tr>
<tr>
<td>tablePadding</td>
<td>integer</td>
<td>The table’s padding value.</td>
</tr>
<tr>
<td>tableSpacing</td>
<td>integer</td>
<td>The table’s spacing value.</td>
</tr>
<tr>
<td>tableTagText</td>
<td>string</td>
<td>Text that contains table tag info that does not have a direct correlation in Fireworks.</td>
</tr>
<tr>
<td>tableWidth</td>
<td>integer</td>
<td>Percentage width if the table in the current cell has a percentage width.</td>
</tr>
<tr>
<td>targetText</td>
<td>string</td>
<td>Target text for this image, if any.</td>
</tr>
<tr>
<td>tdTagText</td>
<td>string</td>
<td>Text of the &lt;td&gt; HTML tag.</td>
</tr>
<tr>
<td>top</td>
<td>integer</td>
<td>Top of the cell in pixels. The top starts at 0.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>Width of the image in pixels, including column spans.</td>
</tr>
</tbody>
</table>
Slices object

Slices is an object that has some properties and is also a two-dimensional array of SliceInfo objects (see “SliceInfo object” on page 284). For example, Slices[0][0] is the slice information 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:

```javascript
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 object properties are read-only.

<table>
<thead>
<tr>
<th>Property (read-only)</th>
<th>Data type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>demoIndex</td>
<td>zero-based index</td>
<td>Index for each file generated for multiple file button export.</td>
</tr>
<tr>
<td>doDemoHTML</td>
<td>Boolean</td>
<td>Corresponds to the Export Multiple Nav Bar HTML Files check box in the Document Specific tab of the HTML Setup dialog box. Setting this property to true produces n+1 HTML pages where n is the number of buttons. A value of false produces a single HTML page.</td>
</tr>
<tr>
<td>doShimEdges</td>
<td>Boolean</td>
<td>Set to true if table shims are set to Transparent Image in Document properties.</td>
</tr>
<tr>
<td>doShimInternal</td>
<td>Boolean</td>
<td>Specifies if the shim mode is internal.</td>
</tr>
<tr>
<td>doSkipUndefined</td>
<td>Boolean</td>
<td>Set to true if Export Undefined Slices is not selected in Document Properties.</td>
</tr>
<tr>
<td>id</td>
<td>integer</td>
<td>Slice ID.</td>
</tr>
<tr>
<td>imagesDirPath</td>
<td>string</td>
<td>Relative URL to the images folder. For example, <em>images/</em>, or <em>./site_images</em>, or <em>&quot;</em> (an empty string) if the images and the HTML are in the same directory.</td>
</tr>
<tr>
<td>layerName</td>
<td>string</td>
<td>Name of the layer.</td>
</tr>
<tr>
<td>menuArrowPath</td>
<td>string</td>
<td>Relative URL to the shim GIF file, arrows.gif.</td>
</tr>
<tr>
<td>menuPath</td>
<td>string</td>
<td>Relative URL of the menu.</td>
</tr>
<tr>
<td>numColumns</td>
<td>integer</td>
<td>Number of columns in the HTML table. Does not include the shim column.</td>
</tr>
<tr>
<td>numRows</td>
<td>integer</td>
<td>Number of rows in the HTML table. Does not include the shim row.</td>
</tr>
<tr>
<td>Property (read-only)</td>
<td>Data type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>shimPath</td>
<td>string</td>
<td>Relative URL to the shim GIF file; for example, images/shim.gif.</td>
</tr>
<tr>
<td>tableNested</td>
<td>Boolean</td>
<td>Specifies if the table is nested.</td>
</tr>
<tr>
<td>width</td>
<td>string</td>
<td>Specifies the slice width.</td>
</tr>
</tbody>
</table>
Chapter 7: Cross-Product Extensions

Cross-product extensions include any Fireworks-related extensions developed for, or in, another Adobe application. These cross-product extensions include those written for other tools, such as Adobe Dreamweaver that use existing Fireworks functionality. They may use JavaScript APIs for adding image-editing functionality to those applications as well as custom Fireworks panels developed in Adobe Flash to enhance the functionality of Fireworks. For example, a developer may want to create an ActionScript command so that a user can replace text in an image without leaving the current movie. Similarly, a Flash developer may create a panel so that a Fireworks user can easily create spirals and other nonstandard shapes repeatedly.

Cross-product architecture

The Fireworks cross-product communication architecture provides a new way for extension developers to create Fireworks-related features for other applications. With this new architecture, your extensions allow a user to perform common image-editing operations (cropping, rotating, adjusting color, blurring, and almost all Fireworks operations) without leaving the current application or opening Fireworks.

XML and remote procedure calls

Applications written with Flash ActionScript 2.0 or C++ applications can control Fireworks by sending JavaScript instructions, called remote procedure calls (RPC), encoded in XML through a local socket. The Fireworks RPC gives other applications access to functionality previously restricted to JavaScript programs running inside Fireworks. The RPC mechanism exposes the Fireworks JavaScript DOM through XML and a TCP socket connection. In this way, an application (written with ActionScript 2.0 or C++) running on the same computer as Fireworks (only local connections are allowed to the loopback address) can be used to open Fireworks documents, slice them, optimize them, and then export them. Users can also create a new Fireworks document through another application, draw in the document, and preview it in the browser. Nearly anything that can be accomplished with a JavaScript program running in Fireworks can now be done through remote procedure calls.

Note: Fireworks excludes functionality related to starting other applications or manipulating non-Fireworks related files (for more information, see “Security” on page 110).
Fireworks RPC transactions pass XML between an RPC client and the Fireworks RPC server built in to Fireworks. The RPC client is any supported program that connects to Fireworks through a TCP stream on port 12124. The Fireworks RPC server is the internal code that listens on TCP port 12124 and then handles client requests. During the RPC transaction, information flows from client to server:

1. Establish TCP connection on port 12124
2. Accept TCP connection
3. Send XML request
4. Process XML request
5. Send XML reply
6. Process XML reply
7. Disconnect TCP connection
8. Accept TCP disconnect

**Note:** The RPC client is not required to disconnect after each XML request. The RPC client can keep the connection open and send additional XML requests (the TCP resubmission timeout is 30 seconds). However, only one XML request can be outstanding at a time. In other words, the RPC client cannot send a second request until it receives a reply to the first request, because there is no queueing mechanism to hold pending requests.

### RPC client XML requests

The XML request contains the following four pieces of information for Fireworks:

- The type of operation to perform
- The name of the operation to perform
- The object on which the operation is performed
- Any parameters the operation needs

**Note:** XML requests are specially formatted XML document fragments, not full XML documents. XML requests are sent to the server in UTF-8 encoding and terminated with the null (0) character.

Logically, requests contain two parts: the envelope and the parameters. The envelope specifies the requested operation (for instance, `get` or `set`) and the object that the operation is performed on. The parameters (strings, integers, arrays, and so on) specify how the operation happens. The envelope tag contains the parameter tag, as follows:

```xml
<envelope><parameter /></envelope>
```

The RPC client sends the following four types of requests, specified in the XML tag name of the envelope:

- The `get` operation retrieves the current object properties. The `get` operation can contain only the `obj` and `name` attributes and no subelements. In the following example, the client requests the value of the `appDir` property of the object whose ID is 1:

```xml
<get obj="1" name="appDir" />
```
The **set** operation sets the object properties. The **set** operation can contain only the **obj** and **name** attributes and exactly one parameter. The parameter must be the same data type as the data type of the property being set, or Fireworks will return an error. In the following example, the client sets the property of “name” (a string providing a directory path) to the value `file://hd/foo/stuff`:

```xml
<set obj="1" name="appDir"><string order="1" value="file://foo/stuff" /></set>
```

The **func** operation calls a method that operates on the specified object. The number and type of parameters vary according to the method called by the **func** operation. In the following example, the client calls the **undo** method to operate on the object with object ID 1:

```xml
<func obj="1" name="undo" />
```

The **release** operation informs the server that the client has finished working on the specified object. A **release** request must specify only an **obj** attribute and no subelements. In the following example, the client tells the server that it has finished working on the object with ID 1:

```xml
<release obj="1" />
```

**Note:** Each type of request requires an **obj** attribute, and all but the **release** request require a **name** attribute. Requests can be only of types **get**, **set**, **func**, or **release**. The RPC server rejects all other types.

### Object IDs

RPC clients reference objects on the server by their object IDs. Because all functionality is exposed by means of objects, every client request must contain a valid object ID. When an object that can be accessed through RPC is created, it is assigned a unique object ID. The object retains that ID for its entire lifetime. This happens for all RPC server objects, whether they are created directly by an RPC call or by an internal Fireworks function. The object IDs can be reused after the corresponding object is destroyed.

**Note:** The object ID number should be treated as a string data type that could contain non-numbers, (do not treat the object ID as an integer data type).

Fireworks has the following four reserved object IDs:

- **"0"**

  This is the Invalid Object ID, used for nonexistent or invalid objects. It is not frequently used for the RPC client, but it is used in several places for the RPC server.

- **"fw"**

  This is the Fireworks Application Object ID. The Fireworks Application Object ID references the main application object in Fireworks and is of the Fireworks class.

  This object is used to open and create documents. In JavaScript, it is the object referenced by `App` or `fw`.

- **"smartShape"**

  This is the SmartShape Object ID. This object id references the global JavaScript variable `smartShape` and is used to create an manipulate Auto Shapes.

- **"Document"**

  This is the Fireworks Document Compatibility Object ID (it is deprecated, like its JavaScript counterpart). It was used in Fireworks 2 for cleaning up file paths, and is included here only for completeness.

- **"Errors"**

  This is the Fireworks Errors Object ID, used mainly for reporting and determining when errors occur in Fireworks. Its JavaScript counterpart is `Errors`.

All other object IDs are generated when the object is created, and may or may not have the same IDs between application invocations.
Data node
The data node is the most important type of XML node in RPC. Methods called through the \texttt{func} operation need to act on actual data or references to server objects identified in data nodes. The data nodes are used as parameters and parts of replies. There are several types of data nodes, as described in the following table.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Node name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>array</td>
<td>&lt;array&gt;&lt;string value=&quot;stuff&quot; /&gt;&lt;int value=&quot;50&quot; /&gt;&lt;/array&gt;</td>
<td>An array data type. It is simply a container node for the other data nodes. There are no restrictions on how many subelements it can contain or which types it can contain. The contained data nodes may be of the same type or of different types. No additional attributes have to be added to the contained nodes.</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
<td>&lt;bool value=&quot;true&quot; /&gt;</td>
<td>The Boolean data type. It can contain either \texttt{true} or \texttt{false} and nothing else. Note that the values are case sensitive.</td>
</tr>
<tr>
<td>dictionary</td>
<td>dict</td>
<td>&lt;dict&gt;&lt;double key=&quot;foo&quot; value=&quot;5.0&quot; /&gt;&lt;string key=&quot;bar&quot; value=&quot;fred&quot; /&gt;&lt;/dict&gt;</td>
<td>A dictionary data type. Like the array data type, it is simply a container for other data nodes. Each direct child node of a dictionary node must contain an additional \texttt{key} attribute. The \texttt{key} attribute is a string and must be unique for the given dictionary node. The \texttt{key} string must start with a letter or an underscore (_), and may followed by numbers, letters, or underscores. Dictionary nodes can be used to pass objects by value.</td>
</tr>
<tr>
<td>float</td>
<td>double</td>
<td>&lt;double value=&quot;1.2345&quot; /&gt;</td>
<td>The floating-point data type. It can contain any floating-point (real) number within the range $1.7e +/- 308$.</td>
</tr>
<tr>
<td>integer</td>
<td>int</td>
<td>&lt;int value=&quot;50&quot; /&gt;</td>
<td>The integer data type. It can contain any signed integer in the range $-2,147,483,648$ through $2,147,483,647$.</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>&lt;null /&gt;</td>
<td>The null type has only one value: null. The null type automatically coerces into the string type, array type, dictionary type, and the server object type. The null type cannot have any attributes or sub-elements.</td>
</tr>
<tr>
<td>server object</td>
<td>obj</td>
<td>&lt;obj value=&quot;fw&quot; class=&quot;Fireworks&quot; /&gt;</td>
<td>The server object data type. The \texttt{value} attribute is set to the object ID. The \texttt{class} attribute is optional. The server always specifies the object class when sending replies to the client. The client, however, is not required to specify the class when sending server object nodes to the server. The \texttt{class} attribute tells the client what properties and methods are available on an object.</td>
</tr>
<tr>
<td>string</td>
<td>string</td>
<td>&lt;string value=&quot;foo&quot; /&gt;</td>
<td>A string data type. It can contain a UTF-8-encoded string, if you include special characters, you must &quot;escape&quot; them (indicate that they are part of the string) according to the method described in the XML Data Model (from <a href="http://www.w3.org/XML/Datamodel.html">http://www.w3.org/XML/Datamodel.html</a>). Most XML writing packages automatically do this.</td>
</tr>
<tr>
<td>void</td>
<td>void</td>
<td>&lt;void /&gt;</td>
<td>No value; no type. Cannot have any attributes or sub-elements.</td>
</tr>
</tbody>
</table>

Parameters
Parameters are simply data nodes with an \texttt{order} attribute. The \texttt{order} attribute identifies the order in which the parameters should be processed for the server. In this way, the RPC client can use any XML client library to build parameters in any order, and the RPC server retains the correct parameter order. The first parameter should have the \texttt{order} attribute set to 0; the second, to 1; and so on, as in this example:

\[
<string order="1" value="bob" />
\]
The `set` operation requires only one parameter, and the `func` operation may have zero or more parameters.

**RPC server XML replies**

After the RPC server processes an XML request, it packages the result as an XML fragment and sends it back to the client. If an error occurred during processing, the server returns an error code in the result XML. Otherwise, the reply node contains a single data node with the result of the operation. If the request doesn’t require a return value, the reply node contains either a single void data node or no children nodes.

For example, here is a successful reply:

```
<return><string value="file://hd/foo/stuff/mydoc.png" /></return>
```

Here is a successful reply with a server object:

```
<return><obj value="23467" class="FireworksDocument" /></return>
```

*Note: When the server returns a server object, it automatically retains the object on the client’s behalf. That is, the object returned to the client is not destroyed until the client releases it with a release request, or until the client disconnects from the server. Therefore, the client should release a server object as soon as the client has finished sending requests related to that object (when the client is done “using” the object).*

**Error codes**

If the server encounters an error when processing a request, the reply node (with the node name `return`) contains at most one `error` attribute, as in this example:

```
<return error="5" />
```

The `error` attribute can contain one of the values listed in the following table.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error occurred, and the request completed successfully. The client should never receive an error attribute with this value. If no error occurred, then no error attribute will be present.</td>
</tr>
<tr>
<td>1</td>
<td>An unknown, generic error occurred. The RPC server could not make enough sense of the request to give a specific error. Check the name of the XML nodes and attributes.</td>
</tr>
<tr>
<td>2</td>
<td>No such object, invalid object ID. The object specified by the client does not exist or the object ID is invalid.</td>
</tr>
<tr>
<td>3</td>
<td>No such method. The method that the client requested does not exist on the specified object.</td>
</tr>
<tr>
<td>4</td>
<td>No such property. The property that the client requested does not exist on the specified object.</td>
</tr>
<tr>
<td>5</td>
<td>Read-only property. The <code>set</code> request cannot be completed because the specified property is read only.</td>
</tr>
<tr>
<td>6</td>
<td>Wrong number of parameters. The request did not specify the correct number of parameters. Either more or fewer parameters are needed.</td>
</tr>
<tr>
<td>7</td>
<td>Wrong parameter type. One or more of the parameters given is of the wrong type.</td>
</tr>
<tr>
<td>8</td>
<td>Security violation. The method is not allowed in RPC.</td>
</tr>
</tbody>
</table>

**RPC and the Fireworks JavaScript DOM**

The RPC server does not allow for self-discovery of the server classes and their methods. Instead the client must know the methods and properties of a given class of objects beforehand. If the client is written in ActionScript or C++, then the client can use the generated client stubs provided by Adobe. Client stubs generated by Adobe know about all methods and properties of every class accessible through RPC. These stubs are available for download at the Adobe website: [http://www.adobe.com/go/learn_fw_documentation_en](http://www.adobe.com/go/learn_fw_documentation_en).
Generating stubs for nonstandard client types

If the client is not written in one of the languages for which Adobe provides a client RPC library, the client implementer must create or generate the stubs. For information about how to do this, see “The Fireworks Object Model” on page 7. The application object (with object ID fw) is of the Fireworks class (for other objects with reserved IDs, see “Object IDs” on page 290). All objects returned by the server contain the class name as an attribute. Given an object’s class, the client can determine what methods and properties it has based on the Fireworks JavaScript DOM. The DOM document also gives the prototypes of the methods and properties of a class. The client can use the DOM document to determine the number and types of method parameters. The DOM document uses more types in its prototypes than the RPC mechanism defines. So, several of the documented types collapse to one RPC type.

In addition to static properties, objects of certain classes can also have dynamic properties. “The Fireworks Object Model” on page 7 documents dynamic properties and specifies whether the dynamic properties are read only. Most dynamic properties are on lists (for example, the behaviorsList class). The properties take an integer or string as a property name, and return a value based on the element associated with the property name.

The following table shows the mapping between the Fireworks Object Model data types and the RPC data types.

<table>
<thead>
<tr>
<th>DOM data type</th>
<th>RPC data type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>array</td>
<td><code>&lt;array&gt;&lt;/array&gt;</code></td>
<td>Types map identically.</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
<td><code>&lt;bool value=&quot;true&quot; /&gt;</code></td>
<td>Both types are identical. Both contain only two values: true or false.</td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td><code>&lt;string value=&quot;#7788CCFF&quot; /&gt;</code></td>
<td>A color is a string with nine characters. It has the format #RRGGBBAA.</td>
</tr>
</tbody>
</table>
| date          | dictionary   | `<dict>`
  `<int key="year" value="2002" />`
  `<int key="month" value="9" />`
  `<int key="day" value="3" />`
  `<int key="hour" value="20" />`
  `<int key="minutes" value="15" />`
  `<int key="seconds" value="32" />` | A date is a dictionary with the following subelement keys: year, month, day, hour, minutes, and seconds. All six elements are integer data types. |
<p>| dictionary    | dictionary   | <code>&lt;dict&gt;&lt;/dict&gt;</code>          | Types map identically.                                                       |
| float         | float        | <code>&lt;double value=&quot;5.132&quot; /&gt;</code> | Types map identically.                                                       |
| integer       | integer      | <code>&lt;int value=&quot;7&quot; /&gt;</code>      | Types map identically.                                                       |</p>
<table>
<thead>
<tr>
<th>DOM data type</th>
<th>RPC data type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>dictionary</td>
<td><code>&lt;dict&gt;</code></td>
<td>A matrix is a dictionary that contains one subelement key: <code>matrix</code>. A matrix is an array of nine float elements. The elements start at the top row and go in row-major order.</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td><code>&lt;null /&gt;</code></td>
<td>Types map identically.</td>
</tr>
<tr>
<td>object</td>
<td>server object, or dictionary</td>
<td><code>&lt;obj value=&quot;1&quot; /&gt;</code></td>
<td>For an object type, the client can simply specify a server object. However, for certain objects (such as objects of the Effect class) a dictionary can be constructed and then used. If the client creates a dictionary, all properties of the class must be added to the dictionary with the correct type.</td>
</tr>
<tr>
<td>point</td>
<td>dictionary</td>
<td><code>&lt;dict&gt;</code></td>
<td>A point is a dictionary with two subelement keys: <code>x</code> and <code>y</code>. Both subelements are float data types.</td>
</tr>
<tr>
<td>rect</td>
<td>dictionary</td>
<td><code>&lt;dict&gt;</code></td>
<td>A rect is a dictionary with four subelement keys: <code>top</code>, <code>left</code>, <code>bottom</code>, and <code>right</code>. All four subelements are float data types.</td>
</tr>
<tr>
<td>resolution</td>
<td>dictionary</td>
<td><code>&lt;dict&gt;</code></td>
<td>A resolution is a dictionary with two subelement keys: <code>units</code> and <code>pixelsPerUnit</code>. The <code>units</code> key can be any of these strings: <code>inch</code>, <code>cm</code>, or <code>pixels</code>. The <code>pixelsPerUnit</code> key is a float data type.</td>
</tr>
</tbody>
</table>
Security

The RPC server restricts some operations to make sure that a client cannot use the RPC server maliciously to damage the user’s system. The first security mechanism is that the RPC server binds to the loopback address, 127.0.0.1. This means all clients must run on the same computer as the RPC server and must connect to that computer through the loopback address. The second security mechanism prevents the exposure of “dangerous” classes that are normally on the Fireworks JavaScript DOM, such as the JavaScript Files classes. However, the client can still have Fireworks open, and export and save PNG and other image files. Third, certain methods and properties that could be used maliciously are not permitted in remote procedure calls (including all methods of the File object, see “Files object” on page 17 and the functions fw.launchApp and fw.findApp). The RPC server returns an invalid method error to the client if it attempts to use these methods or properties. Finally, clients can release only those objects that have been retained on their behalf. Additionally, when a client disconnects, all the server objects that have been retained on that client’s behalf are released.

Note: Objects are not destroyed until the client releases them or until the client that created them disconnects from the server.

RPCMethods class

To use remote procedure calls, Flash developers need to create an instance of a Fireworks object and thereafter manage Fireworks objects carefully to save memory. In ActionScript, you should create blocks of code where you will access the Fireworks DOM, assign a group (pool) of objects to variables, and then release those objects when you are finished. Fireworks provides the following series of memory-management functions to support ActionScript developers working with Fireworks objects. To learn more about using these functions, see “Creating auto-release blocks” on page 297 and “Accessing proxy objects” on page 297. These functions are defined in the supporting RPCMethods.as stubs file available for download from the Adobe website at www.adobe.com/go/fireworks_documentation.

Note: ActionScript remote procedure calls for Fireworks are not ActionScript 1 compatible, and must be written in ActionScript 2.0 (using Flash MX 2004 or later).

RPCMethods.CreateAutoReleasePool()

Usage
RPCMethods.CreateAutoReleasePool()

Arguments
None.

Returns
Nothing.
Description
Starts the auto-release block.

**RPCMethods.DestroyAutoReleasePool()**

**Usage**
RPCMethods.DestroyAutoReleasePool()

**Arguments**
None.

**Returns**
Nothing.

Description
Ends the auto-release block and frees all allocated remote objects in the current auto-release block function.

**RPCMethods.AddToAutoReleasePool()**

**Usage**
RPCMethods.AddToAutoReleasePool(proxyObject)

**Arguments**
- `proxyObject` The object to add to the current pool.

**Returns**
Nothing.

Description
Adds an object to the current auto-release pool. This function is called by the auto-release block. If no object pool exists, this function does nothing.

**RPCMethods.RemoveFromAutoReleasePool()**

**Usage**
RPCMethods.RemoveFromAutoReleasePool(proxyObject)

**Arguments**
- `proxyObject` The object to remove from the current pool.

**Returns**
Nothing.

Description
Removes an object from the current auto-release pool so that it can be used in another code block.
RPCMethods.ReleaseObject()

Usage
RPCMethods.ReleaseObject(Object)

Arguments
Object The name of the object to release from memory.

Returns
Nothing.

Description
Releases an object from memory. This function searches the specified object and all its properties for proxy objects. If proxy objects are found, they are released from memory.

Creating auto-release blocks
Rather than constantly tracking which objects to release and how to release them, you can define blocks of code where you access the Fireworks DOM. In these blocks of code, the ActionScript stubs can keep track of all the proxy objects allocated. Then, at the end of the block of code, a single command will automatically release all objects allocated in the block.

Here is an example:

RPCMethods.CreateAutoReleasePool();
    var selObject = fw.selection.get(0);
RPCMethods.DestroyAutoReleasePool();

Calling the CreateAutoReleasePool() function marks the beginning of the auto-release block, and calling the DestroyAutoReleasePool() function marks the end. Any object allocated between these two calls is released by calling DestroyAutoReleasePool(). Using these two functions, you can write ActionScript code and not worry about memory management.

Accessing proxy objects
If you want to access a proxy object outside of an auto-release block, you must use the RemoveFromAutoReleasePool() function. The RemoveFromAutoReleasePool() function manually removes an object reference from the auto-release pool before exiting the auto-release block.

In this example, the reference to the object defined as selObject is removed:

RPCMethods.CreateAutoReleasePool();
    var selObject = fw.selection.get(0);
RPCMethods.RemoveFromAutoReleasePool(selObject);
RPCMethods.DestroyAutoReleasePool();

Now you can use the selObject object outside the auto-release block.

Note: You must remember to release the selObject object when you are done with it.

Additionally, you can nest auto-release blocks, that is, you can create an auto-release block and then call a function that creates its own auto-release block. The ReleaseObject() function iterates through an object’s properties and releases any proxy objects it finds.
A simple RPC example
This example creates a 200 x 200 pixel rectangle in Fireworks when a button in a Flash application is clicked.

To build a Flash application that uses RPC to create a Fireworks object:
1. Download the supporting ActionScript stub files (a series of supporting ActionScript files) from the Adobe website, you need to put them in your working directory (where the new FLA file will reside).
   You can download the stub for Fireworks 8 from http://www.adobe.com/devnet/fireworks/articles/rpc.html
2. Open a new document in Flash.
3. In the first frame, add the following in the Actions panel to link the general fireworks stub file to the movie when it is published:
   ```
   #include "fwstubs.as"
   ```
4. Create a simple button which will activate the RPC script.
5. Insert the button in the first frame, in the middle of the Stage.
6. Attach the following ActionScript code to the button to activate the RPC code when the button is clicked:
   ```
   on(press){
     RPCMethods.CreateAutoReleasePool();

     var fw = new Fireworks();
     //Hide all panels - this function commented out as it will crash if run from inside Fireworks
     //fw.setHideAllFloaters(true);
     
     //Define the document objects the long way
     var res = new Object();
     res.units = "inch";
     res.pixelsPerUnit = 72;
     var size = new Object();
     size.x = 220;
     size.y = 220;

     //Create new doc
     var fwdoc = fw.createFireworksDocument(size, res, "#0033FF");

     //Define a rectangle object the short way
     var rect = {left:10, top:10, right:210, bottom:210};
     //Add Rectangle
     fwdoc.addNewRectanglePrimitive(rect,0.20);
     //Set its color
     fwdoc.setFillColor("#00CC99");

     RPCMethods.DestroyAutoReleasePool();
   }
   ```
7. Publish the SWF file.
   When you publish the SWF file, make sure the stubs files are in the same directory as your FLA file.
The SWF creates a new Fireworks document, and draws a 200 x 200 pixel green rectangle in Fireworks:

Flash panels

Fireworks contains Adobe Flash Player, which plays Shockwave files as panels and commands in the Fireworks interface. You can also add a Adobe API wrapper extension to Adobe Flash for creating Shockwave files that communicate with the Fireworks API. By leveraging the new API communication between Adobe Flash and Fireworks, Fireworks extension developers can create command interfaces and dialog boxes that go beyond the alert() and prompt() dialog boxes supported in previous versions. You can add command panels to Fireworks for image enhancements, object manipulation, or other custom functionality.

For more information, see Mayur’s article on working with custom panels.

How Flash panels and commands work

Adobe 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, you can embed commands for the Fireworks API in the MMExecute() function. You can construct these Adobe Flash animations as interactive panels that work just as the Layers panel, the Frames panel, and other built-in panels do. For more information, see Mayur’s article.

Shockwave files that are published to the Fireworks installation directory, Configuration\Command Panels folder, act as panels in the Fireworks interface at runtime and are available through the Window menu.

Shockwave files that are published to the Configuration\Commands folder 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 starts Flash Player, which either plays Shockwave animations or runs commands (if the user selects the custom command options). The Align panel (Window > Align menu option) is an example of how Flash panels work in the Fireworks interface.

**Embedding API commands**

You can call any part of the Fireworks API by embedding the API commands in the following functions. These functions communicate directly with Adobe Flash Player, which is distributed with Fireworks.

**MMExecute()**

**Usage**

```javascript
MMExecute(jsToPass)
```

**Arguments**

- `jsToPass` A string of JavaScript for Fireworks to execute.

**Returns**

Nothing.

**Description**

Declares a set of JavaScript code 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.

The commands should be embedded in the same way that you would write separate JavaScript code blocks to perform similar operations, and you can concatenate lines of JavaScript code into one `MMExecute()` function.

**Example**

The following example concatenates two lines of JavaScript code into one command:

```javascript
MMExecute("dom=fw.getDocumentDOM();dom.addNewRectanglePrimitive({left:47, top:26, right:102, bottom:87}, 0);"
```

**MMEndCommand()**

**Usage**

```javascript
MMEndCommand(endStatus, notifyString)
```

**Arguments**

- `endStatus` A Boolean value: true to commit changes; false otherwise. If it is set to false, any pending changes are discarded. To commit the changes, `endStatus` must be set to true.
- `notifyString` A string to notify the user of errors. If the value of `endStatus` is false, this argument holds a string used to notify the user of the error. If `endStatus` is set to true, `notifyString` is an empty string.

**Returns**

Nothing.
Description
This function should be called whenever the user clicks the OK or Cancel buttons provided in the Flash content to execute or cancel a command. This function is used only for modal commands, not for Flash panels.

Note: MMEndCommand supersedes the FWEndCommand command. However, FWEndCommand still works in the current version of Fireworks.

Using the API wrapper extension in Adobe Flash
You can install a special extension that was developed specifically for writing Fireworks functions in ActionScript (currently, only ActionScript 1.0) either as a replacement for the MMExecute() and MMEndCommand() functions or to be used in conjunction with them. After it is installed, the API wrapper appears in the Adobe Flash interface. This wrapper simplifies the writing of Fireworks commands. 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 Adobe Extension Manager installed and double-click the Extension file. In Adobe Flash, the wrapper appears in the Components window as FWCommandComponents.

The following example shows a command without the wrapper:

```javascript
var path = MMExecute("fw.appPatternsDir;");
```

The following example shows the same command using the wrapper:

```javascript
var path =fwapi.getAppPatternsDir();
```

Working with ActionScript files
You can simplify the editing task by keeping a separate ActionScript file for ActionScript; in this way, you don’t need to open and edit the FLA file directly. Your FLA file must have a #include myStringFile.as statement in the first frame (where myStringFile is the name of your ActionScript file) to ensure that the ActionScript strings are complied at publishing time.

Note: The FLA and ActionScript files should reside in the same folder so that the ActionScript file can be easily found for compiling.

Guidelines for creating panels and commands
- You need to surround nested quotation marks need with backslash (\). The following example prints: John’s example is really "complex"!
  ```javascript
  MMExecute('alert("John\'s example is really \"complex\"!");
  ```
- The movie size set in Flash is used in Fireworks as the minimum and default size of the command panel.
- To improve the appearance and positioning of a modeless panel, turn off scaling and align the panel contents with the upper-left corner of the Stage. You can make these changes with the following ActionScript code:
  ```javascript
  Stage.align = "TC";
  Stage.scaleMode = "noScale";
  ```

Events
Fireworks events for Flash panels allow developers to write event handlers for specific user interaction. For example, a panel for creating a customized shape can respond to the user changing the stroke attribute, and make changes to the lines in the shape accordingly.
How event handlers work
When a panel is launched and the Flash movie starts, Fireworks will scan the movie script for the presence of event handlers. If a handler is present, Fireworks automatically registers the function to receive the corresponding event. Fireworks only looks at the SWF file to see if it needs any of these events when the panel opens (when the SWF file runs).

Creating event handlers
To create an event handler, implement a function with the corresponding event name. Currently, Fireworks supports the following events for Flash panels:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onFwStartMovie</td>
<td>Sent to the SWF file right after Fireworks has started (or restarted) the SWF file.</td>
</tr>
<tr>
<td>onFwStopMovie</td>
<td>Sent to the SWF file right before Fireworks stops the file (and possibly unload the file).</td>
</tr>
<tr>
<td>onFwUnitsChange</td>
<td>Sent when the user changes the type of units (inches, pixels, centimeters) in the Info panel.</td>
</tr>
<tr>
<td>onFwPICollapseOrExpand</td>
<td>Sent when the user switches the PI between two rows high and four rows high.</td>
</tr>
<tr>
<td>onFwDocumentNameChange</td>
<td>Sent when the name of the current document changes (for example, when the user performs a save).</td>
</tr>
<tr>
<td>onFwCurrentFrameChange</td>
<td>Sent when the user selects a different frame.</td>
</tr>
<tr>
<td>onFwCurrentLayerChange</td>
<td>Sent when the user selects a different layer.</td>
</tr>
<tr>
<td>onFwHistoryChange</td>
<td>Sent when the user creates a non-scriptable history step.</td>
</tr>
<tr>
<td>onFwIdle0</td>
<td>Sent when Fireworks is in the first of a sequence of idle states. Because Fireworks may often go through a sequence of idle states, triggering functions by this event may impair application performance.</td>
</tr>
<tr>
<td>onFwIdle1</td>
<td>Sent when Fireworks is in the second of a sequence of idle states. Because Fireworks may often go through a sequence of idle states, triggering functions by this event may impair application performance.</td>
</tr>
<tr>
<td>onFwIdle2</td>
<td>Sent when Fireworks is in the third of a sequence of idle states. Because Fireworks may often go through a sequence of idle states, triggering functions by this event may impair application performance.</td>
</tr>
<tr>
<td>onFwApplicationDeactivate</td>
<td>Sent when the Fireworks application loses focus.</td>
</tr>
<tr>
<td>onFwApplicationActivate</td>
<td>Sent when the Fireworks application gains focus.</td>
</tr>
<tr>
<td>onFwSymbolLibraryChange</td>
<td>Sent when the symbol library changes in some way.</td>
</tr>
<tr>
<td>onFwURLListChange</td>
<td>Sent when a new URL is added to the document.</td>
</tr>
<tr>
<td>onFwFavoritesChange</td>
<td>Sent when the favorite URLs list is modified.</td>
</tr>
<tr>
<td>onFwPreferencesChange</td>
<td>Sent when the preferences are changed.</td>
</tr>
<tr>
<td>onFwDocumentOpen</td>
<td>Sent when the document is opened.</td>
</tr>
<tr>
<td>onFwDocumentClosed</td>
<td>Sent when the document is closed.</td>
</tr>
<tr>
<td>onFwDocumentSave</td>
<td>Sent when a save action is performed in the document.</td>
</tr>
<tr>
<td>onFwDocumentSizeChange</td>
<td>Sent when the document is resized.</td>
</tr>
<tr>
<td>onFwActiveViewChange</td>
<td>Sent when the active view changes. This happens when the user changes focus in 2- or 4-Up view.</td>
</tr>
<tr>
<td>onFwPixelSelectionChange</td>
<td>Sent when the pixel selection changes.</td>
</tr>
</tbody>
</table>
Note: The event handler must be implemented in the global namespace. Any events that are in the SWF file but aren’t global, or are read only after Fireworks checks for events, will not work correctly.

Example

```javascript
function onFwDocumentNameChange()
{
    // your code goes here
}
```

Example

```javascript
_global.onFwDocumentNameChange = function ()
{
    // your code goes here
}
```

Both examples show how to implement a handler for the document name changed event. However, the second example will only work if the assignment executes before or during the onFwStartMovie handler.

Publishing

When testing your script, use the File > Publish menu option in Adobe Flash. The Shockwave file is in the same place as the FLA file after publishing.

Debugging

Fireworks provides two functions to help debug Flash panel ActionScript (for more information, see `fw.enableFlashDebugging()` and `fw.disableFlashDebugging()`). Use the Flash debugging functions to show or hide everything that the Shockwave file passes to the Fireworks API during execution. Place these debug functions around the suspect code in your Adobe Flash ActionScript to turn the debugging 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 box statements.
Chapter 8: Auto Shapes

Auto Shapes are vector objects that contain information about how the user can interact with them on the screen. Auto Shapes appear in the Adobe Fireworks user interface as "Auto Shapes" but are programmatically called smartShape objects in the JavaScript code that constructs them. For example, a spiral shape consists of relationships among several smaller objects. A spiral Auto Shape contains additional properties that enable the user to adjust the appearance (stretch, distort, tighten the curve) of the whole spiral by clicking and dragging control points. For more information about user interaction with Auto Shapes, see Using Fireworks Help. You can also find articles about using Auto Shapes on the Fireworks Developer Center at http://www.adobe.com/go/learn_fw_devnet_en.

How Auto Shapes work

You can define an Auto Shape entirely in JavaScript. Auto Shapes installed with Fireworks are located in the Configuration/Auto Shapes folder and the Configuration/Auto Shape Tools folder of the installation directory. The JSF files in this directory contain the JavaScript for each Auto Shape. You can open the files in an editor to see the script for each shape. The Auto Shape file contains a collection of functions that handle the communication between Fireworks and the Auto Shape object (for more information, see "Handling the user interaction" on page 306), define the properties of the Auto Shape, and provide supporting functionality (such as adding other shape objects or performing calculations) as the user manipulates the Auto Shape. The Auto Shape file also contains control points and properties (stroke, fill, color, and so on) that define the shape’s behaviors, appearance, and effects. The points and functions defined in an Auto Shape file use the SmartShape Class and its properties and methods (for more information, see "SmartShape object" on page 272).

Auto Shapes comprise any number of vector objects including open and closed paths and text (currently, nested Auto Shapes are not supported). An Auto Shape can control a bitmap that has been imported into the document; however, Fireworks cannot save a bitmap graphic as an Auto Shape on the user’s drive.

You can create Auto Shape icons for the Tools panel or Auto Shapes panel in PNG, JPG, or GIF format. For the Tools panel, the icon image should be 16 x 16 pixels (if the image is larger than 16 x 16 pixels, Fireworks scales the image to fit in the Tools panel). For the Auto Shapes panel, the Auto Shape icon should be 60 x 60 pixels. If the image for the Auto Shapes panel is smaller, or larger, than 60 x 60 pixels, Fireworks will not scale the image—the icon will appear centered in its cell, but not sized to fit, so it may appear cropped if it is too large.

**Note:** If the icon is missing (or named incorrectly) then Fireworks does not display an icon. However, if the Auto Shapes folder has an icon with the same name as a shape in the Auto Shape Tools folder, then that icon will be used in the Tools panel.

Creating an Auto Shape

To create an Auto Shape, you need to define a series of properties for the shape, define the shape’s control points, and write functions that tell Fireworks how to handle the Auto Shape as the user interacts with the object (for more information, see “Handling the user interaction” on page 306).
Defining the shape

The following code creates the initial shape, a rectangle (a more concise way of creating an initial shape follows this example):

```javascript
function InsertSmartShapeAt()
{
    smartShape.elem.elements[0] = new Path;
    smartShape.elem.elements[0].contours[0] = new Contour;
    smartShape.elem.elements[0].contours[0].nodes[0] = new ContourNode;
    smartShape.elem.elements[0].contours[0].nodes[0].predX = 0;
    smartShape.elem.elements[0].contours[0].nodes[0].predY = 0;
    smartShape.elem.elements[0].contours[0].nodes[0].x = 0;
    smartShape.elem.elements[0].contours[0].nodes[0].y = 0;
    smartShape.elem.elements[0].contours[0].nodes[0].succX = 0;
    smartShape.elem.elements[0].contours[0].nodes[0].succY = 0;
    smartShape.elem.elements[0].contours[0].nodes[1] = new ContourNode;
    smartShape.elem.elements[0].contours[0].nodes[1].predX = 200;
    smartShape.elem.elements[0].contours[0].nodes[1].predY = 0;
    smartShape.elem.elements[0].contours[0].nodes[1].x = 200;
    smartShape.elem.elements[0].contours[0].nodes[1].y = 0;
    smartShape.elem.elements[0].contours[0].nodes[1].succX = 200;
    smartShape.elem.elements[0].contours[0].nodes[1].succY = 0;
    smartShape.elem.elements[0].contours[0].nodes[2] = new ContourNode;
    smartShape.elem.elements[0].contours[0].nodes[2].predX = 200;
    smartShape.elem.elements[0].contours[0].nodes[2].predY = 125;
    smartShape.elem.elements[0].contours[0].nodes[2].x = 200;
    smartShape.elem.elements[0].contours[0].nodes[2].y = 125;
    smartShape.elem.elements[0].contours[0].nodes[2].succX = 0;
    smartShape.elem.elements[0].contours[0].nodes[2].succY = 125;
    smartShape.elem.elements[0].contours[0].nodes[3] = new ContourNode;
    smartShape.elem.elements[0].contours[0].nodes[3].predX = 0;
    smartShape.elem.elements[0].contours[0].nodes[3].predY = 125;
    smartShape.elem.elements[0].contours[0].nodes[3].x = 0;
    smartShape.elem.elements[0].contours[0].nodes[3].y = 125;
    smartShape.elem.elements[0].contours[0].nodes[3].succX = 0;
    smartShape.elem.elements[0].contours[0].nodes[3].succY = 125;
    smartShape.elem.elements[0].contours[0].isClosed = true;
}
```

The Auto Shape is an array of ContourNode objects (see “ContourNode object” on page 246). You can write a “helper” function to simplify the code and assign ContourNode properties, as shown in the following example:

```javascript
function addPathPoint(contour, i, x, y)
{
    var theNodes = contour.nodes;

    // Increase the length to add a new point
    if (i > 0)
        theNodes.length++;

    // get the new point
    var node = theNodes[theNodes.length - 1];

    // Set the new point's values
    node.x = node.predX = node.succX = x;
    node.y = node.predY = node.succY = y;
}
```
You can then simplify the `InsertSmartShapeAt()` function with the new helper function:

```javascript
function InsertSmartShapeAt()
{
    var elem = smartShape.elem;
    var newPath = new Path;
    elem.elements[0] = newPath;
    newPath.contours[0] = new Contour;
    var contour = newPath.contours[0];
    var i = 0;
    addPathPoint(contour, i++, 0, 0);
    addPathPoint(contour, i++, 200, 0);
    addPathPoint(contour, i++, 200, 125);
    addPathPoint(contour, i++, 0, 125);
    contour.isClosed = true;
}
```

**Adding control points**

After selecting an Auto Shape in a document, the user can click its control points to adjust the object. You must define the control points for your Auto Shape before you can define what happens to the object when the user manipulates them.

The following code adds a single control point to the coordinates (0, 0):

```javascript
smartShape.elem.controlPoints.length++;

// Establish the new control point
var cp = smartShape.elem.controlPoints[smartShape.elem.controlPoints.length-1];

// Place the Control Point
cp.x = 0;
cp.y = 0;
```

**Handling the user interaction**

After you define the Auto Shape properties and control points, you need to tell Fireworks how to handle user interactions with the Auto Shape. To facilitate the interaction of the user with the Auto Shape, Fireworks sends a series of messages to the Auto Shape object as the user performs certain operations on the Auto Shape. You can write a series of functions to respond to these messages.

**Fireworks messages**

Fireworks passes the following messages to the SmartShape object as the user interacts with (inserts, moves, or changes) the shape:

- **"InsertSmartShapeAt"**
  Fireworks sends this message when the user selects the shape from the Tools panel and clicks on the canvas, or drags the shape from the Auto Shapes panel to the canvas.

- **"BeginDragInsert"**
  Fireworks sends this message when the user drags an Auto Shape onto the canvas. This message defines a more specific action than the "InsertSmartShapeAt" message.

- **"DragInsert"**
Fireworks sends this message every time the mouse moves during a drag operation (as long as smartshape.getsDragEvents is set to true). For more information, see “SmartShape object” on page 272.

- "EndDragInsert"
  Fireworks sends this message on a mouseUp event after a drag operation.

- "BeginDragControlPoint"
  Fireworks sends this message when the user clicks and holds the mouse button on a control point.

- "DragControlPoint"
  Fireworks sends this message every time the mouse moves during a drag operation (as long as smartshape.getsDragEvents is set to true). For more information, see “SmartShape object” on page 272.

- "EndDragControlPoint"
  Fireworks sends this message when the drag operation is complete.

- "SmartShapeEdited"
  Fireworks sends this message when any change has been made to the Auto Shape (for example, when the user deletes a node).

**Message handler functions**

Because Fireworks sends interaction messages as the user interacts with the Auto Shape, you can write functions to define, edit, and delete the Auto Shape and its properties. Specifically, you write functions defining the effect of manipulating the control points on the shape properties. You can define object properties at various stages of a drag operation: at the beginning of the operation, during the operation, and at its end. (If you define only the properties for the end result, Fireworks waits until the drag operation ends to show the changes to the user.) For example, to have your Auto Shape respond to an “EndDragControlPoint” message, you would write the following function:

```javascript
function EndDragControlPoint(){
    cp.x = smartShape.currentMousePos;
    cp.y = smartShape.currentMousePos;
}
```

The following table lists all the available Fireworks message handler functions you can create (although, you don’t have to write a response to every Fireworks message, only the ones important to your Auto Shape):

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsertSmartShapeAt()</td>
<td>Draws the initial shape. This function is called when the user selects the shape from the Tools panel and clicks on the canvas, or drags the shape from the Auto Shapes panel to the canvas. Define all initial properties of the Auto Shape in this function.</td>
</tr>
<tr>
<td>BeginDragInsert()</td>
<td>Tells Fireworks what to do when the user drags an Auto Shape on the canvas. You can define movements for control points and nodes that you defined in InsertSmartShapeAt().</td>
</tr>
<tr>
<td>DragInsert()</td>
<td>This function is called every time the mouse moves during a drag operation (as long as smartshape.getsDragEvents is set to true). See “SmartShape object” on page 272.</td>
</tr>
<tr>
<td>EndDragInsert()</td>
<td>This function is called on a mouse-up event after a drag operation.</td>
</tr>
</tbody>
</table>
These functions correspond directly with the messages listed in “Fireworks messages” on page 306. To invoke your own function names in response to Fireworks messages, you need to write a switch() statement.

### Switch statements

If you take a look at some existing Auto Shapes (in the Configuration/Auto Shapes folder and in the Configuration/Auto Shape Tools folders), you’ll notice a switch() statement near the beginning of the file. The Auto Shape JavaScript code in these files uses a switch() statement as the initial message handler in the file. The switch() statement sorts the messages sent by Fireworks so each message (that is useful to the particular Auto Shape) invokes a corresponding function.

You can see this switch statement in each of the Auto Shape JavaScript files. Again, a single Auto Shape object may not need to process every message Fireworks sends, so only the useful messages are written into the JavaScript file using the case qualifier. Effectively, the JavaScript file states in case of a certain message, or messages, perform the following function.

In the Frame Auto Shape, this code is used to call PlaceControlPoints() when Fireworks sends a *SmartShapeEdited* message:
switch(smartShape.operation) {
    case "BeginDragInsert":
    case "InsertSmartShapeAt":
        InsertSmartShapeAt(true);
        break;
    case "BeginDragControlPoint":
        BeginDragControlPoint();
        break;
    case "DragControlPoint":
        DragControlPoint();
        break;
    case "EndDragControlPoint":
        EndDragControlPoint();
        break;
    case "SmartShapeEdited":
        PlaceControlPoints();
        break;
}

You don’t need a response for every message Fireworks sends; but you do need to make sure the `switch` statement handles the responses required by your shape.

You can also invoke a single function for several messages:

```javascript
    case "BeginDragInsert":
    case "InsertSmartShapeAt":
        InsertSmartShapeAt(true);
        break;
```

**Supporting functions and methods**

Because the Auto Shape file is written in JavaScript, your functions can use global variables, common functions, and the Fireworks JavaScript API. The Auto Shape JavaScript file contains the definition of the shape’s points, and a series of functions to handle the Fireworks messages as the user interacts with the shape. The file also includes a series of commands and functions defining the shape’s properties and other functionality. These functions are often separate from the message handling functions so they can be used by multiple message handling functions. For example, the Cog Auto Shape JavaScript file (Configuration/Auto Shapes/Cog.jsf) contains user-defined functions near the bottom of the file. These functions perform calculations and create shapes that are useful for the message handling functions. The top of the file contains a series of variable statements that define useful values for tool tips, global variables, and constants used throughout the Auto Shape JavaScript file.

You can use the Fireworks JavaScript API and the Fireworks Object Model, along with efficient JavaScript coding practices, to create effective Auto Shapes (and continue to reuse the most useful functions from each Auto Shape JavaScript file). For more information, see "Additional Fireworks Functions" on page 321 and the "The Fireworks Object Model" on page 7.
Chapter 9: Rich symbols

Rich symbols allow you to create graphic symbols that can be intelligently scaled and given specific attributes using a JavaScript (JSF) file. Quickly mock up a user interface by dragging these symbols onto the document and editing the parameters associated with them using the Symbol Properties panel.

Beginning with Fireworks CS3, you can also export common library assets as known components for use in Adobe Flex™ Builder.™ The MXML export feature allows you to create a Flex application layout in Fireworks, leveraging Flex common library assets as MXML for loading into Flex Builder.

How Rich symbols work

When a symbol is saved as a rich symbol, a PNG file is saved by default in the <user settings>\Application Data\Adobe\Fireworks CS5\Common Library\Custom Symbols folder (Windows), or <user name>/Application Support/Adobe/FireworksCS5/Common Library/Custom Symbols (Macintosh).

To create a rich symbol, a JavaScript file must be created and saved with a .JSF extension in the same location and with the same name as the symbol. For example, mybutton.graphic.png would have a JavaScript file named mybutton.jsf.

The Create Symbol Script panel allows non-programmers to assign some simple symbol attributes and create the JavaScript file automatically. To open this panel, select Create Symbol Script from the Commands menu.

The JavaScript file

Two functions in the JavaScript file must be defined in order to add editable parameters to the symbol:

- function setDefaultValues() -- defines the parameters that can be edited and the default values of these parameters.
- function applyCurrentValues() -- applies the values entered through the Symbol Properties panel to the graphic symbol.

The following is a sample .JSF file for creating a custom symbol:
function setDefaultValues()
{
    var currValues = new Array();
    //to build symbol properties
    currValues.push({name:"Selected", value:"true", type:"Boolean"});
    Widget.elem.customData["currentValues"] = currValues;
}
function applyCurrentValues()
{
    var currValues = Widget.elem.customData["currentValues"];
    // Get symbol object name
    var Check = Widget.GetObjectByName("Check");
    Check.visible = currValues[0].value;
}
switch (Widget.opCode)
{
    case 1: setDefaultValues(); break;
    case 2: applyCurrentValues(); break;
    default: break;
}

This sample JavaScript shows a rich symbol that can change colors:

function setDefaultValues()
{
    var currValues = new Array();
    //Name is the Parameter name that will be displayed in the Symbol Properties Panel
    //Value is the default Value that is displayed when Rich symbol loads first time. In this case, Blue will be the default color when the Rich symbol is used.
    //Color is the Type of Parameter that is displayed. Color will invoke the Color Popup box in the Symbol Properties Panel.
    currValues.push({name:"BG Color", value:"#003366", type:"Color"});
    Widget.elem.customData["currentValues"] = currValues;
}
function applyCurrentValues()
{
    var currValues = Widget.elem.customData["currentValues"]; //color_bg is the Layer name in the PNG that will change colors
    var color_bg = Widget.GetObjectByName("color_bg");
    color_bg.pathAttributes.fillColor = currValues[0].value;
}
switch (Widget.opCode)
{
    case 1: setDefaultValues(); break;
    case 2: applyCurrentValues(); break;
    default: break;
}

To better understand how the .JSF file can be used to customize symbol properties, explore the sample components that have been included with the software.

**Supported attributes for rich symbols**

The following attributes are available in the Create Symbol Script panel (Commands > Create Symbol Script) for customizing rich symbols.
<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>element.left</td>
<td>integer</td>
</tr>
<tr>
<td>element.width</td>
<td>integer</td>
</tr>
<tr>
<td>element.height</td>
<td>integer</td>
</tr>
<tr>
<td>element.pixelRect</td>
<td>rect(left, top, right, bottom)</td>
</tr>
<tr>
<td>element.visible</td>
<td>boolean</td>
</tr>
<tr>
<td>element.opacity</td>
<td>integer</td>
</tr>
<tr>
<td>element.blendmode</td>
<td>string</td>
</tr>
<tr>
<td>element.effectList</td>
<td>object</td>
</tr>
<tr>
<td>element.name</td>
<td>string</td>
</tr>
<tr>
<td>element.mask</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.brushColor</td>
<td>color</td>
</tr>
<tr>
<td>element.pathattrs.fillColor</td>
<td>color</td>
</tr>
<tr>
<td>element.pathattrs.brush</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fill</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.brushTexture</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fillTexture</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fillHandle1</td>
<td>point</td>
</tr>
<tr>
<td>element.pathattrs.fillHandle2</td>
<td>point</td>
</tr>
<tr>
<td>element.pathattrs.fillHandle3</td>
<td>point</td>
</tr>
<tr>
<td>element.pathattrs.setDefaultValue</td>
<td>point</td>
</tr>
<tr>
<td>element.pathattrs.brushPlacement</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fillOnTop</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.pathattributes</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.randSeed</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.textureOffset</td>
<td>point</td>
</tr>
<tr>
<td>element.pathattrs.contours</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.isEvenOddFill</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.fill.category</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.name</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.textureBlend</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.fill.stampingMode</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.edgeType</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.feather</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.fill.ditherColors</td>
<td>object</td>
</tr>
</tbody>
</table>
### Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>element.pathattrs.fill.webDitherTransparent</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.fill.shape</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fill.pattern</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.name</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.nodes</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.opacityNodes</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.color</td>
<td>color</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.position</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.fill.gradient.isOpacityNode</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.fill.pattern.name</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.fill.pattern.image</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.brush.category</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.name</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.angle</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.aspect</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.diameter</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.maxCount</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.minSize</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.softness</td>
<td>?</td>
</tr>
<tr>
<td>element.pathattrs.brush.softenMode</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.shape</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.blackness</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.concentration</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.alphaRemap</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.type</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.feedback</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.flowRate</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.tipCount</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.antiAliased</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.brush.spacing</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.textureBlend</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.textureEdge</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.tipSpacing</td>
<td>integer</td>
</tr>
<tr>
<td>Attributes</td>
<td>Type</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>element.pathattrs.brush.tipSpacingMode</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.tipColoringMode</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.brush.numDashes</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOnSize1</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOffSize1</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOnSize2</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOffSize2</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOnSize3</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.brush.minSize</td>
<td>integer</td>
</tr>
<tr>
<td>element.pathattrs.brush.dashOffSize3</td>
<td>long</td>
</tr>
<tr>
<td>element.pathattrs.mask.element</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.mask.owner</td>
<td>object</td>
</tr>
<tr>
<td>element.pathattrs.mask.linkled</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.mask.enabled</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.mask.mode</td>
<td>string</td>
</tr>
<tr>
<td>element.pathattrs.mask.showAttrs</td>
<td>boolean</td>
</tr>
<tr>
<td>element.pathattrs.mask.autoExpandImages</td>
<td>boolean</td>
</tr>
<tr>
<td>rect.roundness</td>
<td>integer</td>
</tr>
<tr>
<td>rect.pathAttributes</td>
<td>integer</td>
</tr>
<tr>
<td>rect.originalSides</td>
<td>integer</td>
</tr>
<tr>
<td>rect.transform</td>
<td>matrix</td>
</tr>
<tr>
<td>text.antiAliased</td>
<td>boolean</td>
</tr>
<tr>
<td>text.antiAliasMode</td>
<td>string</td>
</tr>
<tr>
<td>text.autoKern</td>
<td>boolean</td>
</tr>
<tr>
<td>text.orientation</td>
<td>string</td>
</tr>
<tr>
<td>text.pathAttributes</td>
<td>object</td>
</tr>
<tr>
<td>text.randSeed</td>
<td>argLong</td>
</tr>
<tr>
<td>text.textChars</td>
<td>string</td>
</tr>
<tr>
<td>text.textureOffset</td>
<td>point</td>
</tr>
<tr>
<td>text.transformMode</td>
<td>string</td>
</tr>
<tr>
<td>text.rawTop</td>
<td>integer</td>
</tr>
<tr>
<td>text.rawLeft</td>
<td>integer</td>
</tr>
<tr>
<td>text.rawWidth</td>
<td>integer</td>
</tr>
<tr>
<td>text.rawHeight</td>
<td>integer</td>
</tr>
</tbody>
</table>
MXML Export

The Flex application framework consists of MXML, ActionScript 3.0, and the Flex class library. Developers use MXML to declaratively define the application user interface elements and use ActionScript for client logic and procedural control. Developers write MXML and ActionScript source code using the Adobe Flex Builder® IDE or a standard text editor.

MXML is the XML-based language that developers use to lay out components in Flex applications. It provides a declarative approach to controlling an application’s visual appearance. Using MXML, developers can position components and specify constraints to enable a more “liquid” user interface. Developers can also use View States and Transitions to control how the application should respond to different user actions and application events.

Using Fireworks, you can export Flex code (MXML) in order to easily create a Flex application layout in Fireworks, leveraging Flex common library assets, and export it as MXML for loading into Flex Builder.

To export MXML data
1. Choose File > Export.
2. Choose MXML and Images in the Export pop-up menu.
3. Select the Put images in subfolder option if you want to save the images in a separate folder from the MXML code.
4. Select the Current page only option to export only the currently selected page.
5. Click Save to complete the export.

Flex Component Definitions for MXML and Images Export

When exporting a Fireworks document with the MXML and Image export, Fireworks checks each element’s customData property for definitions that dictate how that element should appear in the resulting MXML. If no MXML-related definitions are found, the element is exported as an image. Each Flex component found in the Fireworks common library already has unique customData definitions that allow it to be exported as MXML. If you wish to have elements within your document export as their own custom MXML tags you can do so with the following customData definitions.

flexClassName

Flex component symbols are associated as being Flex Components by the MXML and Images import through a customData property called flexClassName.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>text.autoExpand</td>
<td>boolean</td>
</tr>
<tr>
<td>text.italic</td>
<td>boolean</td>
</tr>
<tr>
<td>text.underline</td>
<td>boolean</td>
</tr>
<tr>
<td>text.bold</td>
<td>boolean</td>
</tr>
<tr>
<td>text.justify</td>
<td>string</td>
</tr>
<tr>
<td>text.font</td>
<td>string</td>
</tr>
<tr>
<td>text.fontsize</td>
<td>integer</td>
</tr>
<tr>
<td>text.alignment</td>
<td>string</td>
</tr>
</tbody>
</table>
Format

```
flexClassName = "WhichClass";
```

When this property is present in a Fireworks element, the MXML and Images export process exports the element as a specific MXML tag and not as an image. Its value indicates the tag name of the MXML class generated.

Example

```
element.customData["flexClassName"] = "ComboBox";
```

The above line indicates that the element object is a Flex component whose MXML class is ComboBox. For rich symbols this code should be placed within the `setDefaultValues` block of code (`Widget.opCode == 1`) that initially defines the rich symbol.

Example

```
function setDefaultValues() {
    Widget.elem.customData["flexClassName"] = "ComboBox";
    // additional code...
}
switch (Widget.opCode) {
    case 1: setDefaultValues(); break;
    // additional code...
}
```

All Flex widgets provided with Fireworks already have this property defined. The class name used determines the behavior of each of those components as determined internally within the MXML and Images export.

When you create a custom rich symbol, you must define the `flexClassName` property if you want the MXML and Images export to treat the rich symbol as an MXML tag and not an exported image. The class name you decide to use will be used in the resulting MXML tag. For example, if you defined `flexClassName` as "Foo" the resulting MXML tag is `<mx:Foo .../>`.

**flexClassDefinition**

In addition to the `flexClassName` property, there is an additional, optional `customData` property `flexClassDefinition`, which can be used to further influence the resulting MXML export.

Format

```
flexClassDefinition = classDefinitionObject;
```

Example

```
classDefinition = new Object();
element.customData["flexClassDefinition"] = classDefinition;
```

A `flexClassDefinition` object includes any number of the following properties, all of which are themselves optional.

**margin**

Rectangle. Defines the area around the element that is not part of the Flex component shape but takes up space in Fireworks (for example, the Panel component in Flex has a drop shadow that is not included as part of the panel’s size in Flex but is included when recreated in Fireworks). It consists of four integer properties measured in pixels: top, right, bottom, and left.
Example

```javascript
margin = {top:5, right:5, bottom:5, left:5};
```

### padding

Rectangle. Defines the area within the element that determines if another element can be seen as being a child of that element. If defined, any other element whose bounds are completely within the area marked by padding and arranged above the current element will be a child element within the resulting MXML. It consists of four integer properties measured in pixels: top, right, bottom, and left; and is calculated after the margin.

Example

```javascript
padding = {top:5, right:5, bottom:5, left:5};
```

![Diagram showing margin and padding relationships in respect to a panel component](image)

### sizeOffset

Object. Defines a size offset to apply to the MXML definition of the exported element. It consists of two integer properties, width and height, which are measured in pixels and are added to the MXML width and height attributes. This does not have any affect on margin or padding properties. It simply consists of values that will be added to the final width and height attributes of the MXML tag produced.

Example

```javascript
sizeOffset = {width:10, height:10};
```

### customValues

Object. Contains custom name-value pairs that are available to the MXML export but not accessible as visible properties within Fireworks. During export these are translated directly to MXML attributes or styles in combination with any properties defined within `customData currentValues`.

Example

```javascript
customValues = {myNum:10, click:"clickHandler()"};
```
attributeProperties
Array. A list of property names that will become attributes in the exported MXML tag. By default, all rich symbol properties (defined as `currentValues` or `customValues`) are considered attribute properties. By defining this list, you can restrict which properties are used as attributes. Those not provided will not be included as attributes of the resulting MXML tag.

Example
```
attributeProperties = ["label", "enabled"];  
```

styleProperties
Array. A list of property names which are to be defined in the CSS style tag for this class. If any of these properties are also attribute properties (implicit or explicit), they will instead become style properties.

Example
```
styleProperties = ["color", "textSelectedColor"];  
```

defaultProperties
Array. A list of the names of automatic attribute properties created for the MXML tag. Automatic attributes are those properties which are not listed as `currentValues` or `customValues` but are automatically determined and generated by the MXML export. This includes the following properties:

- x: x position of the element on the screen
- y: y position of the element on the screen
- width: width of the element
- height: height of the element
- alpha: transparency of the element
- id: ID tag (based on the element name)
- source: source location of the image for objects exported as images
- styleName: reference to the style that applies to this element, if applicable

If this list is not defined, all default properties are added to the attribute list (where applicable). If defined, only those listed will be provided.

**Note:** `defaultProperties` cannot be style properties. Omitting `styleName` from `defaultProperties` will not prevent a style from being created for the object if it has style properties.

Example
```
defaultProperties = ["x", "y", "width", "height"]; 
```

namespace
Object. The namespace to be used in the MXML tag. This object should contain one key representing the namespace prefix whose value is the namespace URI. If this is not provided, the default namespace of `mx` is used.

Example
```
namespace = {ns:"http://www.example.com/2007/mxml"}; 
```
textOnly
String. If defined, this represents the entire MXML output to be used for the export. If this is defined, no other properties need to be set within the definition. Any properties set are ignored.

Example
textOnly = "<mx:ColorPicker id="cp" selectedColor="#FFFFFF"/>;"

ignored
Boolean. If true, the object is completely ignored during export; no MXML is created and no image is exported. If this is set to true, no other properties need to be set within the definition. Any properties set are ignored.

Note: The ignored property has precedence over textOnly.

Example
ignored = true;

Example script
The following is an example of a custom symbol script using a custom definition object to determine the output created by the Fireworks MXML and Images export.

```
function setDefaultValues() {
    var currValues = new Array();
    currValues.push({ name:"name", type:"Text", value:"star name" });
    currValues.push({ name:"label", type:"Text", value:"Star" });
    currValues.push({ name:"color", type:"Color", value:"#FFFFFF" });
    currValues.push({ name:"enabled", type:"Boolean", value:"true" });
    Widget.elem.customData["currentValues"] = currValues;
    Widget.elem.customData["flexClassName"] = "Star";
    var classDef = new Object();
    classDef.margin = {top:0, right:3, bottom:10, left:3};
    classDef.padding = {top:30, right:10, bottom:10, left:10};
    classDef.sizeOffset = {width:10, height:0};
    classDef.customValues = {staticValue:"static"};
    classDef.attributeProperties = ["staticValue", "name"];
    classDef.styleProperties = ["color"];
    classDef.defaultProperties = ["x","y","width","height"];
    classDef.namespace = {ns:"http://www.example.com/2007/mxml"};
    Widget.elem.customData["flexClassDefinition"] = classDef;
}
function applyCurrentValues() {
    var vals = Widget.elem.customData["currentValues"];
    Widget.GetObjectByName("label").textChars = vals[1].value;
    Widget.GetObjectByName("label").pathAttributes.fillColor = vals[2].value;
    Widget.GetObjectByName("disabled").visible = !vals[3].value;
}
switch (Widget.opCode) {
    case 1: setDefaultValues(); break;
    case 2: applyCurrentValues(); break;
}
```

A document containing this rich symbol that is exported using the MXML and Images export would result in the following MXML document.
With the definition of `flexClassName`, the element in Fireworks exports as a `Star` MXML tag. Notice how the properties defined within `attributeProperties` became attributes and the property (color) defined within `styleProperties` became part of the `Star` style. The only default properties created were those defined in `defaultProperties`; `x`, `y`, `width`, and `height`. Because a namespace was defined, the tag was given the correct prefix and its URI was defined within the `Application` tag.

In Fireworks, the star in this example was 147 x 138 but exported with a width and height of 157 x 138 in MXML because of `sizeOffset`. Though not apparent in this example, margin and padding would also play a part in how this element is exported.
Chapter 10: Additional Fireworks Functions

This chapter lists additional JavaScript functions supported by Adobe Fireworks that let you create useful Fireworks extensions and customized Fireworks menus. Almost any task that the user can accomplish in Fireworks with the menus, tools, or floating panels can be done programmatically using JavaScript.

Property inspector functions

These functions control the Property inspector window, which shows details about the current document or selected object.

fw.showPIWindow()

Availability
Fireworks MX.

Usage
fw.showPIWindow()

Arguments
None.

Returns
Nothing.

Description
Opens the Property inspector.

fw.hidePIWindow()

Availability
Fireworks MX.

Usage
fw.hidePIWindow()

Arguments
None.

Returns
Nothing.
Description
Makes the Property inspector window invisible.

**fw.isPIExpanded()**

**Availability**
Fireworks MX.

**Usage**
fw.isPIExpanded()

**Arguments**
None.

**Returns**
A Boolean value: true if expanded; false otherwise.

Description
Determines whether the Property inspector window is currently expanded or minimized

**fw.isPIVisible()**

**Availability**
Fireworks MX.

**Usage**
fw.isVisible()

**Arguments**
None.

**Returns**
A Boolean value: true if visible; false otherwise.

Description
Determines whether the Property inspector window is currently hidden or shown.

**fw.growPIWindow()**

**Availability**
Fireworks MX.

**Usage**
fw.growPIWindow()
Arguments
None.

Returns
Nothing.

Description
Expands the Property inspector window.

fw.shrinkPIWindow()

Availability
Fireworks MX.

Usage
fw.shrinkPIWindow()

Arguments
None.

Returns
Nothing.

Description
Minimizes the Property inspector window.

fw.setPIPosition()

Availability
Fireworks MX.

Usage
fw.setPIPosition(pt)

Arguments
pt A point in screen coordinates.

Returns
Nothing.

Description
Moves the upper-left corner of the Property inspector window to the specified location.
**fw.getPIPosition()**

**Availability**
Fireworks MX.

**Usage**
fw.getPIPosition()

**Arguments**
None.

**Returns**
A point object that is formatted as \{x:float, y:float\} (see “Point data type” on page 9 for syntax details), which contains the location of the Property inspector.

**Description**
Retrieves the location, in screen coordinates, of the upper-left corner of the Property inspector window.

### History panel functions

These functions control the History panel.

**fw.historyPalette.clearSteps()**

**Availability**
Fireworks 3.

**Usage**
fw.historyPalette.clearSteps()

**Arguments**
None.

**Returns**
Nothing.

**Description**
Clears the undo and redo stack.

**fw.historyPalette.copySteps()**

**Availability**
Fireworks 3.
Usage
fw.historyPalette.copySteps(array of indexes)

Arguments
array of indexes  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.

Description
Copies history steps to the Clipboard.

fw.historyPalette.getSelection()

Availability
Fireworks 3.

Usage
fw.historyPalette.getSelection()

Arguments
None.

Returns
A zero-based array that represents which History panel steps are selected.

Description
Determines which steps in the History panel are selected.

fw.historyPalette.getStepCount()

Availability
Fireworks 3.

Usage
fw.historyPalette.getStepCount()

Arguments
None.

Returns
The number of steps in the History panel (not a zero-based value).

Description
Gets the number of steps in the History panel.
fw.historyPalette.getStepsAsJavaScript()

Availability
Fireworks 3.

Usage
fw.historyPalette.getStepsAsJavaScript(array of indexes)

Arguments
array of indexes 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.

Description
Gets the JavaScript equivalent of the specified steps.

See also
fw.historyPalette.replaySteps()
/fw.historyPalette.getUndoState()

Availability
Fireworks 3.

Usage
fw.historyPalette.getUndoState()

Arguments
None.

Returns
The string to use with fw.historyPalette.setUndoState().

Description
Returns a string that indicates the current undo state to be used for later calls to 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().

See also
fw.historyPalette.setUndoState()
fw.historyPalette.replaySteps()

Availability
Fireworks 3.

Usage
fw.historyPalette.replaySteps(array of indexes)

Arguments
array of indexes 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.

Description
Gets the JavaScript equivalent of the specified steps and executes them.

See also
fw.historyPalette.getStepsAsJavaScript()

fw.historyPalette.saveAsCommand()

Availability
Fireworks 3.

Usage
fw.historyPalette.saveAsCommand(array of indexes, {filename})

Arguments
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 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 /Configuration/Commands folder with the specified filename.

Returns
Nothing.

Description
Gets the JavaScript equivalent of the specified steps and saves them as a JSF command file.
fw.historyPalette.setSelection()

Availability
Fireworks 3.

Usage
fw.historyPalette.setSelection(array of indexes)

Arguments
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.

Description
Sets the portion of the History panel that is selected.

fw.historyPalette.setUndoState()

Availability
Fireworks 3.

Usage
fw.historyPalette.setUndoState(undoStateString)

Arguments
undoStateString The string returned by fw.historyPalette.getUndoState().

Returns
Nothing.

Description
Performs the correct number of undo or redo operations to arrive at the selected state.

See also
fw.historyPalette.getUndoState()

Static Document Functions

Fireworks has an older static Document object that has been deprecated in favor of the Document object in the DOM. The static Document object is only accessed by two methods. You can access this API using document.methodName().
**document.findExportFormatOptionsByName()**

**Availability**
Fireworks 3.

**Usage**
document.findExportFormatOptionsByName(name)

**Arguments**
name 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.

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

**document.makeGoodNativeFilePath()**

**Availability**
Fireworks 3.

**Usage**
document.makeGoodNativeFilePath(fileURL)

**Arguments**
fileURL 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.

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

**Example**
The following command returns "file:///My Documents/image01.png".
document.makeGoodNativeFilePath("file:///My Documents/image01.png")