About Flash Asset Xtra for Flash Player 8

Acknowledgments

Project Management: Stephanie Gowin

Writing: Jody Bleyle

Editing: Mark Nigara

Media Design and Production: Adam Barnett

Special thanks to Tom Higgins

First Edition: December, 2005

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

Contents

About Flash Asset Xtra for Flash Player 8

Macromedia Xtra extensions are software components that add features to Macromedia Director. Flash Asset Xtra for Flash Player 8 lets you add a Macromedia Flash Player 8 sprite to a Director movie and exposes Flash ActionScript objects to Lingo and JavaScript syntax. Flash Player 8 is a major player release that includes huge performance improvements, support for image effects (such as filters and blending modes), improved text rendering, JavaScript integration, file input and output, and a new video codec (On2 VP6).

Please refer to the Macromedia Flash documentation for a complete list of Flash Player 8 features. This document, *About Flash Asset Xtra for Flash Player 8*, provides an overview of Flash Player 8 features, calls out a few outstanding features, lists any unsupported features, and documents a new API—`convert()`—that can be accessed as both a top-level function and a Flash sprite method in Lingo and JavaScript syntax.

System requirements

The following hardware and software are the minimum required to author Director movies and run Macromedia Flash Asset Xtra software:

- For Microsoft Windows: An Intel Pentium III 600 MHz (or later processor running Windows 2000 or Windows XP; at least 128 MB of available RAM (256 MB recommended); and 200 MB of available disk space
- For Macintosh: A Power Macintosh G3 500 MHz (or later) running Mac OS X 10.2.6 or 10.3; at least 128 MB of available RAM (256 MB recommended); and 200 MB of available disk space

The following hardware and software are the minimum required to play Director movies:

- For Microsoft Windows: An Intel Pentium II with 64 MB of available RAM running Windows 98, or an Intel Pentium III with 128 MB of available RAM running Windows 2000 or Windows XP; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.01 Service Pack 2, Microsoft Internet Explorer 5.5 Service Pack 2, or Microsoft Internet Explorer 6 Service Pack 1
- For Macintosh OS X: A Power Macintosh G3 with 128 MB of available RAM running Mac OS X 10.1.5, 10.2.6, or 10.3; one of the following web browsers: Netscape 7.1, Microsoft Internet Explorer 5.2 or later, or Safari 1.1

NOTE	There is no Macintosh Classic version of Flash Asset Xtra for Flash Player 8.

Installing Flash Asset Xtra and Flash Asset Options Xtra

Follow these steps to install Flash Asset Xtra and Flash Asset Options Xtra on a Windows or Macintosh computer.

NOTE	The Flash Asset Options Xtra is an authoring-only Xtra that provides support for certain dialog boxes within Director.

To install Flash Asset Xtra and Flash Asset Options Xtra:

1. Unzip the Flash Asset Xtra and Flash Asset Options Xtra files.
2. Close Director if it is open.
3. Place the Xtra extensions in the following location:
   - For Windows: c:\Program Files\Macromedia\Director MX 2004\Configuration\Xtras\Media Element\Flash Asset
   - For Macintosh: /Applications/Macromedia Director MX 2004/Configuration/Xtras/Media Element/Flash Asset

   An Xtra extension can be stored in a folder up to five folders below the Xtras folder.
4. Open Director.

NOTE	Copies of the same Xtra extension can have different filenames or they may have the same filename but may reside in different folders. If duplicate Xtra extensions are available when Director starts, Director displays an alert. Delete any duplicate Xtra extensions.

Guide to instructional media

The Flash Asset Xtra package includes this document, *About Flash Asset Xtra for Flash Player 8,* which helps you learn the program quickly and become proficient in using Flash sprites in your Director movies.

Additional resources

For more information about Flash Asset Xtra, see Changes to the Macromedia Flash Asset Xtra in Director MX 2004.

For the latest information on Flash Asset Xtra, plus advice from expert users, advanced topics, examples, tips, and other updates, see the Macromedia Director Developer Center website, which is updated regularly.

For the latest information on Flash Player 8, see the Macromedia Flash Player Developer Center website, which is updated regularly.

For TechNotes, documentation updates, and links to additional resources in the Director and Flash Communities, see the Macromedia Support Center.

Typographical conventions

The following typographical conventions are used in this book:

- *Italic font* indicates a value that should be replaced (for example, in a folder path).
- `Code font` indicates code (ActionScript, Lingo, or JavaScript).
- *`Code font italic`* indicates a code parameter.

What's new in Flash Asset Xtra

Flash Asset Xtra for Flash Player 8 provides compatibility with Flash Player 8 expressiveness features affecting bitmaps and vector shapes. A new Director API, convert(), lets you convert object types between Lingo and ActionScript.

This section provides the following information:

- An overview of the new features in Flash Player 8 (adapted from the Macromedia Flash Developer Center article Migrating from Flash MX 2004 to Flash 8).
- A list of ActionScript classes that are new in Flash Player 8 and supported in the Xtra.
- A list of Flash Player 8 features that are not supported in Flash Asset Xtra. All Flash Player 8 features are supported in the Flash Asset Xtra unless they are listed here.

For a complete list of what's new in Flash Player 8, see the section What's New in Flash 8 ActionScript > New in ActionScript 2.0 and Flash 8 > Additions to the ActionScript language in *Learning ActionScript 2.0 in Flash* in Flash Help or in LiveDocs format.

For complete Flash Player 8 API documentation, see *ActionScript 2.0 Language Reference* in Flash Help or in LiveDocs format.

New Flash Player 8 features

This section highlights some of the new features that have been added to ActionScript 2.0 in Flash 8.

On2 VP6 video codec lets you encode video files using the On2 VP6 video codec, which provides superior video quality using the smallest possible file size.

Alpha channel support lets you encode video in which Flash removes and saves the background as a transparency. This allows you to overlay (or composite) the video on top of other Flash content, while the subject of the video remains in the foreground. For example, you can use alpha channels to record a video clip of a presenter using a blue screen as a backdrop. You can then encode the video with an alpha channel and place the presenter in front of another image as the backdrop to the video.

Bitmap filter effects let you use ActionScript to apply filters to objects that Flash Player 8 renders at runtime. The filters include drop shadow, blur, glow, bevel, gradient glow, and gradient bevel. You can also use an adjust color filter that lets you edit a movie clip's brightness, contrast, saturation, and hue.

The drawing API lets you control the line style of strokes that you draw and allows you to create more complex gradients to fill shapes.

Text handling improvements let you use ActionScript to access advanced anti-aliasing features such as FlashType. In addition, Macromedia has added new options, properties, and parameters to the TextField and TextFormat classes.

Load new kinds of image files at runtime, such as progressive JPEG images and non-animated GIF and PNG files. If you load an animated file, the first frame of that animation appears in the SWF file.

Bitmap caching lets you improve the performance of your applications at runtime by caching a bitmap representation of your vector button or movie clip instances at runtime. You can use ActionScript code to access this property or select the instance and then select an option in the Property inspector to cache the instance. Caching a movie clip as a bitmap prevents Flash Player from having to redraw the image continually, providing a significant improvement in playback performance.

Alpha channel masking is supported if both the mask and the maskee movie clips use bitmap caching. This support also lets you use a filter on the mask independently of the filter that is applied to the maskee itself.

NOTE	Mask layers do not support alpha channel masking. You must use ActionScript code to apply a mask and use runtime bitmap caching.

9-slice scaling enables you to scale movie clip instances without widening the strokes that outline the movie clip. You can use ActionScript code or the Flash user interface to access this feature.

The Flash Asset Xtra 8 supports 9-slice scaling of movie clips within Flash 8 sprites. 9-slice scaling lets you scale movie clip instances without widening the strokes that outline the movie clip. You can use ActionScript code or the Flash user interface to access this feature in either Flash Basic 8 or Flash Professional 8.

A new garbage collector is built into Flash Player. It uses an incremental collector to improve performance. For more information, see the Macromedia Flash Developer Center article Performance Improvements in Flash Player 8.

New Flash Player 8 ActionScript classes

The following table lists classes and language elements new or changed in Flash 8 and supported in Flash Player 8 and Director. For more information on these classes and language elements, see the Flash documentation.

Class	Package	Description
`BevelFilter`	flash.filters	Adds bevel effects to objects
`BitmapData`	flash.display	Creates and manipulates arbitrarily sized transparent or opaque bitmap images
`BitmapFilter`	flash.display	Acts as a base class for filter effects
`BlurFilter`	flash.filters	Applies blurs to objects in Flash
`ColorMatrixFilter`	flash.filters	Applies transformations to ARGB colors and alpha values
`ColorTransform`	flash.geom	Adjusts color values in movie clips (Color class is deprecated in favor of this class)
`ConvolutionFilter`	flash.filters	Applies matrix convolution filter effects
`DisplacementMapFilter`	flash.filters	Uses pixel values from a BitmapData object to perform displacement on an object
`DropShadowFilter`	flash.filters	Adds drop shadows to objects
`FileReference`	flash.net	Lets you upload and download files between the user's computer and a server.
`FileReferenceList`	flash.net	Lets you select one or more files to upload.
`GlowFilter`	flash.filters	Adds glow effects to objects
`GradientBevelFilter`	flash.filters	Adds gradient bevels to objects
`GradientGlowFilter`	flash.filters	Adds gradient glow effects to objects
`IME`	in the System class	Manipulates the operating system's input method editor (IME) within Flash Player
`Locale`	mx.lang	Controls how multilanguage text appears in a SWF file
`Matrix`	flash.geom	Represents a transformation matrix that determines how to map points from one coordinate space to another

Class	Package	Description
`Point`	flash.geom	Represents a location in a two-dimensional coordinate system (x represents the horizontal axis and y represents the vertical axis)
`Rectangle`	flash.geom	Creates and modifies Rectangle objects
`TextRenderer`	flash.text	Provides functionality for anti-aliasing embedded fonts
`Transform`	flash.geom	Collects data about color transformations and coordinates manipulations that you apply to a MovieClip instance

NOTE

Official support is added for the AsBroadcaster class in Flash 8. This class was not officially supported in Flash MX 2004.

NOTE

Macromedia has also added new language elements, methods, and functions to existing ActionScript classes. For more information, see *Learning ActionScript 2.0 in Flash* in Flash Help.

Unsupported Flash Player 8 features

The following Flash Player 8 features are not supported in the Flash Asset Xtra for Flash Player 8.

- Version 2 components
- Accessibility code
- ExternalInterface class

Using Flash objects in ActionScript

ActionScript classes are organized in *packages*. A package is a directory that contains one or more class files and resides in a designated classpath directory. A package can, in turn, contain other packages, called subpackages, each with its own class files. Packages are commonly used to organize related classes. For example, the BitmapFilter class and the BlurFilter class are classes that allow you to create visual effects. The following classes belong to the `filters` package: `flash.filters.BitmapFilter` and `flash.filters.BlurFilter`.

To use a class in Flash ActionScript, you can import the package and then instantiate the object, as follows:

```
// Import the package file.
import flash.filters.BlurFilter;

// Instantiate that object.
var tMyFilter:BlurFilter = new BlurFilter(...);
```

You can also use the full package name to instantiate an object and import the package in the same line of code, as follows:

```
var tMyFilter:flash.filters.BlurFilter = new flash.filters.BlurFilter(...);
```

To use an ActionScript object in Director, you must use the full package name. The following Lingo examples instantiate a BlurFilter class:

```
// Instantiate a global Flash object.
tMyFilter = newObject("flash.filters.BlurFilter", ...)

// Instantiate a Flash object within a Flash sprite.
tMyFilter = sprite("SWF").newObject("flash.filters.BlurFilter", ...)
```

However, if you know that a Flash sprite is playing a SWF file in which Flash has imported the `flash.filters` package, you can refer to the class directly in Lingo and JavaScript syntax, as follows:

```
tMyFilter = sprite("SWF").newObject("BlurFilter", ...)
```

NOTE

You cannot refer to an ActionScript object directly with global Flash objects because the global Flash Player instance cannot import package files.

Publishing Flash content for Flash Asset Xtra

The Flash Asset Xtra wraps Flash Player 8 and therefore supports the same backward compatibility as Flash Player 8. Flash content published as any Flash Player version (Flash Player 1-8) should play correctly in Flash Player 8.

If you want Flash Asset Xtra to properly execute and display content that contains Flash Player 8 features, you must publish the Flash content as Version 8.

Updating Shockwave movies to use Flash Asset Xtra

If you have a movie with a version 8 SWF file that you are publishing for playback in Shockwave in a browser, you need to prepare the movie so that the updated Xtra gets downloaded to the client's computer. When the updated movie is played on a system that does not have the latest version of the Flash Asset xtra, the Shockwave Player will automatically download and install the latest version so the movie plays correctly.

To update a Shockwave movie to use Flash Asset Xtra:

1. Open the DIR file to update in Director MX 2004.
2. Choose Modify > Movie > Xtras.
3. In the Movie Xtras dialog, select one of the following:
   - (Windows) Flash Asset.x32
   - (Macintosh) Flash Asset PPC
4. Deselect the Download if Needed check box.
5. Check the Download if Needed check box.

   Director retrieves the package file from the server and stores the version information in the movie.
6. Click OK to close the Movie Xtras dialog box.
7. Choose File > Save to save the movie.
8. Choose File > Publish Settings and publish a Shockwave (DCR) file.

To check the updated Shockwave movie:

Enter the following code in the Message window:

```
put the movieXtralist
```

The Flash Asset version information displays in the Message window.

Creating Flash Video content for Flash Asset Xtra

The On2 VP6 codec provides superior video quality using the smallest possible file size. You can use Flash Asset Xtra to deliver high-quality digital video in a Director movie without requiring an external video player.

There are many options for creating video in Flash. Flash Player 8 even lets you include alpha channels in your videos. For more information, see Working with Video in *Using Flash* in Flash Help or in LiveDocs format.

NOTE	You cannot directly import or insert Flash Video (FLV) files into Director. FLV files must play inside a SWF file that is imported into Director.

About ActionScript image effects

Several new Flash Player 8 ActionScript classes help you create powerful image effects. Flash Asset Xtra lets you access these classes from Director instead of building your own effect routines in Lingo.

If you want to move image data between Lingo and ActionScript for rendering purposes, you must use the Lingo convert() API to convert the image to the correct data type. This section provides examples of two workflows that use the Flash Asset Xtra capabilities with the `convert()` API to create image effects.

NOTE	If you render the image in its native environment (for example, an ActionScript image in a Flash sprite or a Lingo image in Director), you don't need to convert the image data type to access ActionScript classes from Lingo if you want to create image effects.

The following example takes a Lingo image, converts it to an ActionScript BitmapData object, applies an ActionScript blur effect to it, and renders it in Director as a 3D texture:

```
-- Get a Lingo image.
tImage = member("SomeImage").image.duplicate()

-- Create an ActionScript BlurFilter object. Note that the filter's
-- properties are default values. To specify non-default values,
-- pass additional parameters in the newObject() call.
tBlurFilter = newObject("flash.filters.BlurFilter")

-- Convert the image object to a BitmapData object.
tBitmapData = convert(#bitmapdata, tImage)

-- Apply the blur filter to the BitmapData object.
-- Trap the returned result value for error checking.
tResult = tBitmapData.applyFilter(tBitmapData, tBitmapData.Rectangle,
  newObject("Point",0,0), tBlurFilter)

-- Use the returned result value to error check.
-- A value of zero indicates that no error occurred.
-- Negative values indicate that an error occurred.
if (tResult = 0) then

-- Convert the blurred BitmapData object back to a Lingo image object
  tBlurryImg = convert(#image,tBitmapData)

-- Use the image object to create a new texture and apply it to a model in
  the scene
  tTexture = member("3D").newTexture("Box",#fromImageObject,tBlurryImg)
  member("3D").model("Box").shader.texture = tTexture
end if
```

The following example pushes a Lingo image into a Flash sprite, converts the image to an ActionScript BitmapData object, and renders the image in Flash:

```
-- Get a Lingo image
tImage = member("Some Image").image.duplicate()

-- Convert the image object to a BitmapData object
-- inside the Flash sprite that will render the image
tBitmapData = sprite("SWF").convert(#bitmapdata, tImage)

-- Attach the BitmapData object to a movie clip.
-- (Note: this example assumes that on the Stage in the SWF
-- is a movie clip instance named "MovieClip".)
-- (Note: there are additional optional parameters that
-- can be specified in the call to attachBitmap().)
sprite("SWF").MovieClip.attachBitmap(tBitmapData,1)
```

ActionScript BitmapData objects can begin to consume a lot of memory while waiting for Flash memory garbage collection to occur. To avoid excess memory usage, call the `dispose()` method to flush individual BitmapData objects from memory, as in the following:

```
-- Get a Lingo image object.
tImg = member("foo").image

-- Convert it to a BitmapData object and do some filter effects.
tBMP = convert(#bitmapdata,tImg)

-- Insert filter manipulation code here.

-- Convert it back to an image object
-- and dispose of the BitmapData object.
tImg = convert(#image,tBMP)
tBMP.dispose()
```

Translating image data types between Lingo and ActionScript

The Flash Asset Xtra adds one new Lingo API, the convert() method, which converts image data types between Lingo and ActionScript formats. This allows you to use new Flash Player 8 ActionScript classes to manipulate bitmap and vector images more expressively and display them in a Director movie. It also allows you to render Lingo images in Flash sprites.

For more information, see "About ActionScript image effects" on page 14.

About image data types

To use the ActionScript filter effect objects or drawing routines on an image, you must ensure that the image is an ActionScript object. You must convert Lingo image data to an ActionScript BitmapData object to apply any filter effects, use the ActionScript drawing API, or render the image in a SWF file.

Conversely, an image must be a Lingo image object to be rendered in Director or to be manipulated using Lingo imaging commands. For example, if you want to use Flash to add a blur effect to a Lingo image object, you must convert the Lingo image to an ActionScript BitmapData object, manipulate it in ActionScript, and convert it back to a Lingo image object.

convert()

Usage

-- Lingo syntax

```
convert(targetDataFormat, sourceDataRef)
sprite(whichSprite).convert(targetDataFormat, sourceDataRef)
```

// JavaScript syntax

```
convert(targetDataFormat, sourceDataRef);
sprite(whichSprite).covert(targetDataFormat, sourceDataRef);
```

Description

Top-level function or Flash sprite method. Translates image, rectangle, and list/array objects between Lingo (Director) and ActionScript (Flash). Returns a reference to the converted object. For example, in the following code, `x` is a valid reference to an ActionScript Rectangle value:

```
x = newObject("flash.geom.Rectangle")
```

In the following code, `y` is a a valid reference to a Lingo rect value:

```
y = convert(#rect, z)
```

The following table lists the equivalent Lingo and ActionScript objects:

Lingo data type	ActionScript data type
image	BitmapData
rect	Rectangle
list	Array

NOTE The `convert()` method does not operate on points. To convert an ActionScript Point object to a Lingo Point object, pass the ActionScript Point object to the Lingo `point()` function. To convert a Lingo `point` variable to an ActionScript Point object, use the `newObject()` function.

NOTE These ActionScript data types are also called classes. They are documented in the ActionScript classes section of the *ActionScript 2.0 Language Reference* in Flash Help.

Parameters

targetDataFormat Required. A symbol value indicating the target data format for the conversion process. The following values are supported: `#image` (Lingo image object), `#rect` (Lingo rect object), `#bitmapdata` (ActionScript BitmapData object) and `#rectangle` (ActionScript Rectangle object).

sourceDataRef Required. A reference to the source data to be converted. This parameter must be a reference to a Lingo image object, a Lingo rect object, an ActionScript BitmapData object or an ActionScript Rectangle object.

Example

The following statements convert a BitmapData object in a Flash sprite to a Director image object so that the image can be used as a texture within a 3D world :

--Lingo syntax

```
tBmpData = sprite(12).getVariable("ImageData", false)
tImageObj = sprite(12).convert(#image, tBmpData)
tTexture = member("3D").newTexture("Flash
  texture",#fromImageObject,tImageObj)
```

// JavaScript syntax

```
var tBmpData = sprite(12).getVariable("ImageData", false);
var tImageObj = sprite(12).convert(symbol("image"), tBmpData);
tTexture = member("3D").newTexture("Flash
  texture",#fromImageObject,tImageObj);
```

The following statements convert the Director rect value to a Flash Rectangle object in the global Flash-object space:

--Lingo syntax

```
tImageRect = member("Bitmap").image.rect;
tRectangle = convert(#rectangle, tImageRect);
```

// JavaScript syntax

```
var tImageRect = member("Bitmap").image.rect;
var tRectangle = convert(symbol("rectangle"), tImageRect
```

The `convert()` method only converts linear Lingo lists and normal Flash arrays; it does not convert property lists, or associative arrays. Also, the `convert()` method only operates on Lingo lists, not on JavaScript arrays.

The following statements convert arrays and lists:

--Lingo syntax

```
trees = ["oak", "ash", "elm", "beech"]
flashtrees = convert(#flashObjectArray, trees)
put flashtrees.toString()
```

```
-- "oak,ash,elm,beech"
put flashtrees[0]
-- "oak"
```

// JavaScript syntax

```
trees = list("oak", "ash", "elm", "beech");
flashtrees = convert(symbol("flashObjectArray"), trees);
trace (flashtrees.tostring()); // oak,ash,elm,beech
trace(flashtrees[1]); // ash
```

--Lingo syntax

```
bikes = newObject("Array", "trek", "giant", "schwinn", "colnago")
lingobikes = convert(#list, bikes)
put lingobikes
-- ["trek", "giant", "schwinn", "colnago"]
put lingobikes[1]
-- "trek"
```

// JavaScript syntax

```
bikes = newObject("Array", "trek", "giant", "schwinn", "colnago");
lingobikes = convert(symbol("list"), bikes);
trace(lingobikes.toString()); // <["trek", "giant", "schwinn", "colnago"]>
trace(lingobikes[1]); // trek
```

When setting an array property of a component sprite or member, lists are now properly converted to Flash arrays. For example, if sprite 1 is a List component, you can use the following code to assign the `labels` property:

--Lingo syntax

```
sprite(1).labels = ["uno", "dos", "tres", "quatro", "cinco"]
```

// JavaScript syntax

```
sprite(1).labels = list("alpha", "beta", "gamma", "delta", "epsilon");
```

Additional Lingo APIs

Macromedia added two additional Flash sprite APIs in Director 10.1: `propDirectAccess` and `createVariable()`.

NOTE

The only other new Flash sprite API is the convert() function (this API is also a top-level function), which is covered in the section "About ActionScript image effects" on page 14.

propDirectAccess

Usage

```
spriteObjRef.propDirectAccess
```

NOTE	Macromedia added this property in Director version 10.1.

Description

Flash sprite property; controls whether you can use dot syntax to directly access ActionScript variables with a sprite reference. If the value is 1, you can access ActionScript variables with dot syntax; if the value is 0, you cannot. The default value is 0 for movies created with all versions of Director.

Example

The following code creates a variable called `aPie` and gives it the value `3.14159`. The subsequent statements can use dot syntax to access the value of the variable.

-- Lingo syntax

```
sprite(1).createVariable(#aPie, 3.14159)
put sprite(1).propDirectAccess
-- 1
put sprite(1).aPie
-- 3.1416
sprite(1).propDirectAccess = 0 put sprite(1).aPie
-- This code generates a "property not found" script error.
```

// Javascript syntax

```
sprite(1).createVariable(symbol("aPie"), 3.14159);
trace(sprite(1).propDirectAccess);
// 1
trace(sprite(1).aPie);
// 3.14159
sprite(1).propDirectAccess = 0;
trace(sprite(1).aPie);
// [function aPie]
```

createVariable()

Usage

-- Lingo syntax

```
spriteObjRef.createVariable(varNameSymbol, varValue)
```

// JavaScript syntax

```
spriteObjRef.createVariable(varNameSymbol, varValue);
```

NOTE	This function was added in Director version 10.1.

Description

Flash sprite command; creates an ActionScript variable in the Flash sprite. This command is related to the `getVariable()` and `setVariable()` Flash sprite functions.

Parameters

`varNameSymbol` Required. Specifies the name of the variable to create. This is passed in as a Lingo symbol; the string representation for the symbol is used as the variable name.

`varValue` Required. Specifies the initial value for the new variable.

Example

The following statement creates a variable called `bill` and gives it the value `5`:

-- Lingo syntax

```
sprite(1).createVariable(#bill, 5)
put sprite(1).bill
-- 5
```

// Javascript syntax

```
sprite(1).createVariable(symbol("bill"), 5);
trace(sprite(1).bill);
// 5
```

This example first creates an ActionScript array, and then creates the variable `lisa` to store the array within the Flash sprite.

-- Lingo syntax

```
temp = sprite(1).newObject("Array", "one", "two", 3, 4)
sprite(1).createVariable(#lisa, temp)
put sprite(1).lisa[1]
-- "two"
```

// Javascript syntax

```
temp = sprite(1).newObject("Array", "one", "two", 3, 4);
sprite(1).createVariable(symbol("lisa"), temp);
trace(sprite(1).lisa[1]);
// two
```