

A Meaningful Conversation Between JavaScript and Flash

By Phillip Kerman

Overview

If you think Flash authoring software has limited programming attributes, think again. Anything JavaScript can do, Flash can, too. Flash can send and receive messages to JavaScript, making it the logical step for any developer who wants to go beyond the basic features of HTML and Flash. Flash includes certain fundamental limits which make JavaScript the best solution to many challenges.

Why bother?

Learning a new programming language (JavaScript) is an investment. The knowledge you gain, however, is applicable elsewhere. JavaScript's "document object model" is identical to Director 7's "dot syntax". Also, the relationship of a browser *hosting* a Flash movie is similar to Director or Authorware *hosting* Flash, Authorware *hosting* Director, and Dreamweaver *hosting* a custom behavior.

Before adopting JavaScript consider using Flash's built-in features. Through the action *Tell Target*, Flash can "talk" to a movie clip. Also, *Load Movie* provides a way to modularize your movie into individual files. These two actions can make Flash movies very interactive... and require nothing more than the built-in Flash features.

When the fundamental limit of Flash is reached JavaScript can provide solutions. The crux of this limit lies in variables and tracking. Although future versions of Flash may overcome part of this limit, JavaScript is a true programming language with multiple data types, extensive functions, and, most importantly direct access to the browser (including functions available today like "cookies", as well as the means to reach future controls whenever they're introduced).

What can it do?

A Flash movie (via the FSCommand action) can invoke a JavaScript function which, in turn, can evaluate an expression or execute a function. Think of this as *Flash talking to JavaScript*. For example, a user could click a button in a Flash movie which launches a JavaScript "prompt" from which the user selects their language... the result (their answer) is then saved in a cookie for this site.

JavaScript can "talk" to Flash in one of two basic ways. JavaScript can "tell" a Flash movie (or movie clip) to do something, like *go to* a particular frame. Also, JavaScript can "ask" (and get) information from Flash, such as *what is the current frame?*

Sequence of Events:



1. The FSCommand action placed in a frame or under a button.
2. The function named `movieObj_DoFSCommand()` is invoked. Where "movieObj" matches the *name* (in the embed tag) and the *ID* (in the object tag). (`NAME=movieObj ID=movieObj`)
3. JavaScript will execute any expressions inside this function.
4. Regardless of how many instances of the FSCommand action are in the Flash movie, the same DoFSCommand function is invoked. However, each instance can be separated by including a different "Command" in the field (named *command*) in the FSCommand action. Additional parameters can also be included. The JavaScript function changes to `movieObj_DoFSCommand(command)` and the value for *command* can be ascertained in JavaScript:

```
if (command=="this") { //do this } else if (command=="that") { //do that }
```



1. A JavaScript function is called. (Inside another function or as a result of an event such as "onClick".)
2. The Flash object is referred to (See *Netscape/IE Differences* below).
Then, either an action occurs or information is returned to JavaScript.

Examples of actions occurring:

<code>flashObject.play()</code>	→ Flash Movie Plays
<code>flashObject.stopPlay()</code>	→ Flash Movie Stops

Example of information returned to JavaScript: `someVariable=flashObject.currentFrame()` → Flash returns the current frame number...(and that number is placed into `someVariable`)

Netscape/IE Differences:

Embedding. Flash movies are “embedded” into Netscape and contained in an “object” tag in Internet Explorer. In either case, your Flash object/embed needs a name. `ID=anyNameForObject (IE)` and `NAME=anyNameForObject(Netscape)`.

Object Reference. To control Flash from JavaScript, you simply refer to the Flash object and indicate which function you want to execute. Like “`object.play()`”. However, to refer to that object, IE uses `window[objName]` and Netscape uses `document[objName]`. If you only have one Flash movie embedded, it’s preferable to refer to the object (with Netscape) as `document.embeds[0]` (zero is the first thing embedded). Of course, if the Flash movie to which you want to refer is in another window or frame, you can precede this reference with the path to the object—like: `top.frames[0].document.embeds[0]`;

Redirect. A Flash movie “talking” to JavaScript in IE won’t work unless you redirect it *to* the `FSCCommand`. It’s pretty simple, there’s a standard VBSCRIPT subroutine which can “listen” for Flash and then *call* the `FSCCommand`. As it turns out many browsers don’t respond well to this VBSCRIPT—so the preferred method is to store the entire script in a string variable and then “write” it into your document with a `document.write(aString)`;

Funkiness:

0,1,2,3... Flash’s first frame is “frame 1”. JavaScript, and HTML for that matter, always count by starting with zero. For example, JavaScript can tell a Flash movie to go to its first frame with `document[flashobj].GoToFrame(0)`;

Instance Hierarchy. JavaScript can talk to a Flash movie’s movie clip instance directly by name. However, the name needs to be preceded by “_flash0”. `TPlay('_flash0/movieClipInstanceName')`; or `TPlay('_flash0/clip/clipInAClip')`;

Kick Start. When JavaScript tells a Flash movie to go to a frame, it acts like the “Go To” action. Adding an additional JavaScript `Play()`; will make the action perform like a “Go To and Play”.

Wait for it... JavaScript can only talk to a Flash movie which is entirely downloaded. Otherwise, results will be unpredictable or fail. There are strategies to avoid this issue. If Flash initially invokes the JavaScript (that is, Flash first tells JavaScript to do something which requires JavaScript to talk to Flash), simply use the “if frame is loaded” action in Flash to assure the whole movie is downloaded before initiating the JavaScript. However, if JavaScript will start the communication (say, after an “onLoad” or “onClick” event) then use `document[flashObj].PercentLoaded()` to determine if the Flash movie has reached 100% download.

Workarounds. There are two simple JavaScript functions which don’t work properly within the `FSCCommand` (or any JavaScript originally invoked by Flash). Namely, `location.href` and `window.open()`. The workaround is to use `setTimeout()`. For example, `setTimeout("location.href= 'other.html'",0)`; will change the location to “other.html” in 0 seconds.

Special thanks and credit is well deserved by the following:

TimeTrek Game:

Spooky and the Bandit -
an interactive brand of Think New Ideas.
www.spookyandthebandit.com

Clock example 1:

ENS—Engineering Network Services (www.enetserve.com)

Clock example 2 and information:

John Croteau www.flashtek.com

Clock example 3 and ‘Merger’ game:

Van Phan www.mergergames.com

Information:

Dave Livesay

Graphics and editing:

Diana Bauer

Editing:

Sheri Kramer Macromedia

Downloads:

This document, presentation, links, and downloads

[www.teleport.com/~ phillip/ucon99/presentation/](http://www.teleport.com/~phillip/ucon99/presentation/)

Resources / References:

Bertoflash:

www.bertoflash.nu/

Flash Zone:

www.flashzone.com/

Flasher ListServ:

http://www.chinwag.com/html/mailling_lists_0.html

Flashpad:

www.flasher.net/flashpad.html

Getting Started with FSCCommand:

www.erols.com/dlivesay/FSCCommand.html

JavaScript Demo:

www.macromedia.com/support/flash/how/subjects/javascriptdemo/

Scripting with Flash:

www.macromedia.com/support/flash/how/subjects/scriptingwithflash/

The Flash Academy:

www.enetserve.com/tutorials/

JavaScript: The Definitive Guide and *JavaScript Pocket Reference*

by David Flanagan, O’Reilly, ISBN: 1-56592-392-8 and 1-56592-521-1

JavaScript for Dummies

by Emily A. Vander Veer, IDG Books, ISBN: 0-7645-0112-7

A Meaningful Conversation Between JavaScript and Flash, 26 May 1999
Macromedia User Conference 1999

Phillip Kerman

+ 1 503 236-7721

phillip@teleport.com

www.teleport.com/~phillip