



Adobe Flex-Ajax Bridge (FABridge) Cheatsheets

The Flex-Ajax Bridge (FABridge) is a code library used to expose the Flash Player to JavaScript. Typical usage involves the existence of Flex applications embedded in JavaScript/DHTML applications.

Materials List

Item	Description	Where to get it	Required
Flex 2 SDK	Free software development kit used to develop Flex applications.	http://www.adobe.com/products/flex/sdk/	Yes
FABridge.js	JavaScript side of the bridge.	1. Part of the LiveCycle Data Services download: http://www.adobe.com/products/livecycle/dataservices/ 2. skipslate example download	Yes
FABridge.as	ActionScript side of the bridge.	1. Part of the LiveCycle Data Services download: http://www.adobe.com/products/livecycle/dataservices/ 2. skipslate example download	Yes
Flash Player Content Debugger	Used to enable your JavaScript to capture more useful exception data from ActionScript.	http://www.adobe.com/support/flashplayer/downloads.html	No, but if you do install it, your apps are exposed to more exception handling data.

Recommended Directory Structure

The following is a recommended directory structure. The only hard requirement here is that the *FABridge.as* file be located in a directory named *bridge*. This *bridge* directory must reside at the same level as your *.mxm*l source file at compilation. The reason for this is that the *FABridge* class is defined in the *bridge* package.

```
<app_root>
+--- assets
|   +--- js
|   |   +--- bridge
|   |   |   +--- FABridge.js
|   |   +--- <other_scripts...>.js
|   +--- swf
|       +--- bridge
|       |   +--- FABridge.as
|       +--- <some_mxml>.mxml
|       +--- <some_swf>.swf
+--- index.html
```

Script Installation

To install the *FABridge.js* script, simply link it in to your HTML.

```
<html>
<head>
    <title>FABridge Example</title>
    <script language="JavaScript" src="assets/js/bridge/FABridge.js"></script>
... rest of html ...
```

To install the *FABridge.as* script to your .mxml source file, you'll need to define the component namespace and link it in. When you link the component in, you'll also give it a *bridgeName* to reference in your JavaScript. The example below defines the *bridge* namespace and links all components which reside in the *bridge* directory which exists at the same level as the .mxml source file.

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns:bridge="bridge.*">
    <bridge:FABridge bridgeName="flex" />
... rest of mxml
```

Common Task #1 – Creating the Flex Application Instance in JavaScript

Since you have to wait for the SWF to load before you can access it, it is best to create a callback to fire after initialization. This example is accessing a Flex application assigned the *bridgeName* of **flex**.

```
// global variable
var flexApp;

var initCallback = function() {
    flexApp = FABridge.flex.root();
    return;
}
// register the callback to our Flex application with the 'bridgeName' of 'flex'
FABridge.addInitializationCallback( "flex", initCallback );
```

Common Task #2 – Get Objects by Id, Get Object Properties

Given the following *TextInput* control in an .mxml source file:

```
<mx:TextInput id="txt_mainDish" text="Chicken Yaya" />
```

Accessing the *text* property of this control is as follows:

ActionScript:

```
txt_mainDish.text
```

JavaScript (FABridge):

```
// get the TextInput object by id
var swfObj_mainDish = flexApp.getTxt_mainDish();
// get the text property of the TextInput object
var txt_mainDish = swfObj_mainDish.getText();
```

Notice that when accessing object id's and properties using JavaScript and the FABridge, they're called like functions using the following format:

```
<flex_app_instance>.get<id>( )
```

```
<actionscript_object>.get<property>( )
```

After the *get* in this notation, also note the alternate capitalization naming convention.

Common Task #3 – Setting Object Properties

Given the following *TextInput* control in an .mxml source file:

```
<mx:TextInput id="txt_mainDish" text="Chicken Yaya" />
```

Setting the text property of this control is as follows:

ActionScript:

```
txt_mainDish.text = "cheeseburger";
```

JavaScript (FABridge):

```
// get the TextInput object by id
var swfObj_mainDish = flexApp.getTxt_mainDish();
// set the text property of the TextInput object
var txt_mainDish = swfObj_mainDish.setText( "cheeseburger" );
```

The notation is similar to *get* as follows:

```
<actionscript_object>.set<id_or_propertyname>( <value_to_set> )
```

Common Task #4 – Calling Object Methods

Object methods may be directly accessed like they are in ActionScript.

```
// get the TextInput object by id
var swfObj_mainDish = flexApp.getTxt_mainDish();
// bring focus to the TextInput object
swfObj_mainDish.setFocus( );
```

Common Task #5 – Passing Event Handlers to ActionScript

FABridge also allows us to attach event handlers to ActionScript objects residing in the Flash player. Given the following MXML:

```
<mx:TextInput id="txt_mainDish" text="Chicken Yaya" />
```

and the following HTML:

```
<form name="frm_menu">
  <input type="text" name="frm_mainDish" value="" />
  ... rest of form ...
```

We can attach a *change* event listener to the *txt_mainDish* ActionScript object. The example event listener below will update the value of the *frm_mainDish* HTML text input control as the value in the ActionScript *txt_mainDish* control changes.

```
var swfObj_mainDish = flexApp.getTxt_mainDish();
var mainDishCallback = function( event ) {
```

```

    // get the object which fired the event
    var swfObj_source = event.getTarget();
    // as the value in the swf 'Main Dish' changes, so does the html 'Main Dish'
    document.frm_menu.frm_mainDish.value = swfObj_source.getText();

    return;
}
swfObj_mainDish.addEventListener( "change", mainDishCallback );

```

Common Task #6 – Exception Handling

Exceptions occurring on the ActionScript side are caught and thrown over the bridge to JavaScript. This can be accounted for with a try/catch block in JavaScript wrapped around any potentially offending code. If you install the content debugger for Flash player, you have access to more useful information. Otherwise you just get an error code and you'll have to look up the description yourself.

```

var str_msg = "";

try {
    flexApp.potentialOffender()
}
catch( e ) {
    // message is supported without the content debugger
    str_msg += "Name: "      + e.name      + "\n\n";
    str_msg += "File Name: " + e.fileName + "\n\n";
    str_msg += "Line Number: " + e.lineNumber + "\n\n";
    str_msg += "Message: "   + e.message  + "\n\n";
    str_msg += "Stack: "     + e.stack    + "\n";

    alert( str_msg );
}

```

Useful Links

Compiler Error Listing: <http://livedocs.adobe.com/flex/201/langref/compilerErrors.html>

Compiler Warning Listing: <http://livedocs.adobe.com/flex/201/langref/compilerWarnings.html>

Run-Time Error Listing: <http://livedocs.adobe.com/flex/201/langref/runtimeErrors.html>