Using FlexJS with Adobe Flash Builder

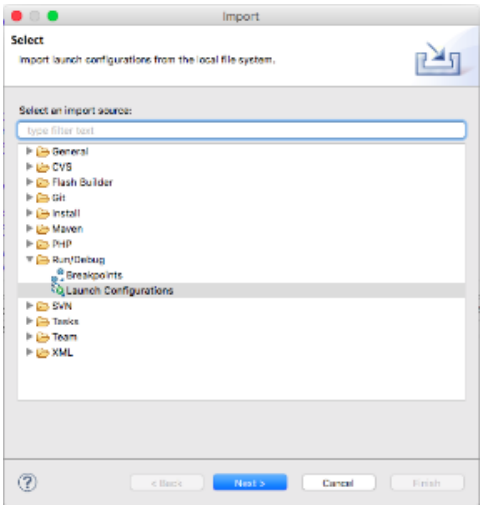
Follow these instructions to use Flash Builder 4.7 for developing FlexJS applications. These instructions assume you have installed the Apache FlexJS SDK via one of the methods described in "Getting Started".

Also, starting with Apache FlexJS 0.8.0, you will need to have upgraded Flash Builder to run on Java 7 or 8. See http://blogs.adobe.com/flashplayer/2016/09/running-adobe-flash-builder-on-mac-with-java-78.html

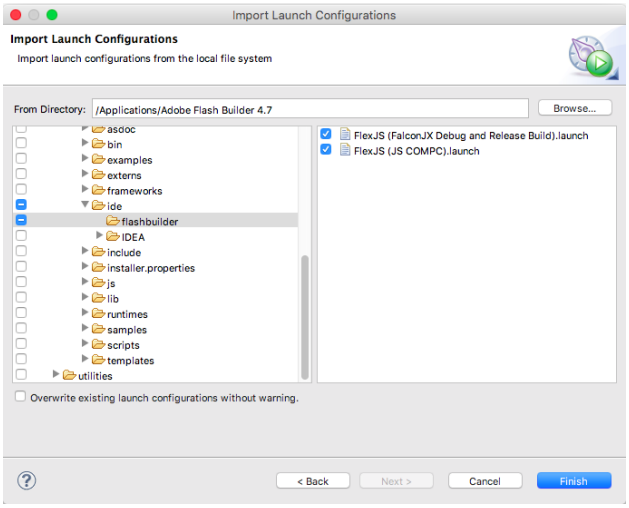
Install FlexJS SDK as a Flex SDK

The FlexJS SDK is designed to look like the Flex SDK so that Flash Builder can just treat it as the regular Flex SDK. To add the FlexJS SDK to Flash Builder:

1. Run Flash Builder
2. In the Flash Builder Preferences menu add this new folder as a Flex SDK
3. Choose from the File menu, Import, Run/Debug, Launch Configurations and click the Next button



4. Browse to the FlexJS SDK directory and in the "/ide/flashbuilder" folder you will see new launch configurations. Select these and click the Finish button

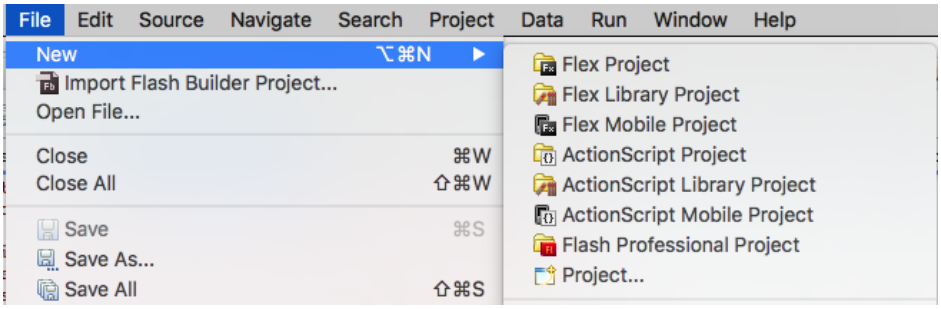


There should now be around 9 new configs in the Run menu under External Tools

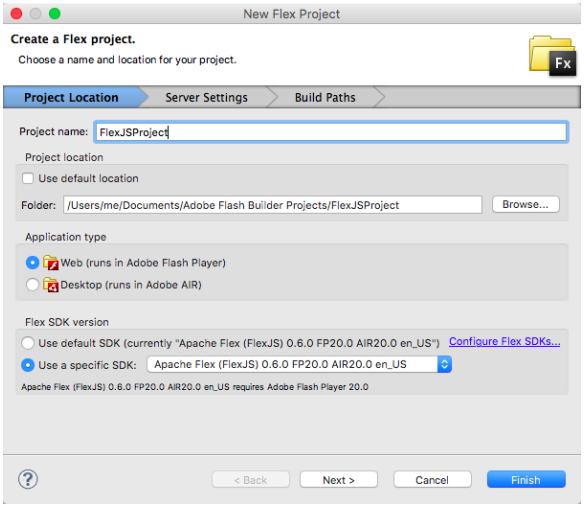
Creating a FlexJS Application

Flash Builder does not have a default Application template for a FlexJS project. The recommended way to create a new Flex JS application is to create a Flex Project specifying the FlexJS SDK and then use a launch config to modify the project. Another alternative is to copy in an existing working example and modify it as follows:

1. Create a new Flex Project from the Flash Builder File menu



2. Choose the Apache FlexJS SDK and click Finish or Next to change and default options



   Flash Builder will generate a new project and a new Application MXML file that has errors. Ignore those errors for now.

   If you create a Desktop (AIR) project, you will probably get an error in the error log that says: "Error while retrieving predefined variables". Just ignore that for now.

3. You can then choose the "Convert New Flex Project to FlexJS Project" script from the list of External Tools in the Run menu. This will replace files in the new project to look like the Hello World example below. You can instead choose the "Convert New Flex Project to FlexJS MVC Project" script and it will replace files in the new project to look more like the DataBindingExample below.

   Or you can copy the example code below into your Application MXML file or skip to the next section to start with the example project

Hello World:

Hello World

```
<?xml version="1.0" encoding="utf-8"?>
<js:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
                xmlns:js="library://ns.apache.org/flexjs/basic"
                >

    <fx:Style>
        @namespace basic "library://ns.apache.org/flexjs/basic";

        .labelStyles {
            font-weight: bold;
            background-color: blue;
        }
    </fx:Style>

    <!-- Initial View -->
    <js:initialView>

        <js:View>
            <js:Label text="Hello World" x="100" y="100" className="labelStyles"/>
        </js:View>

    </js:initialView>

    <js:valuesImpl>
        <js:SimpleCSSValuesImpl />
    </js:valuesImpl>

</js:Application>
```

Full Example Project

For a more comprehensive example locate the DataBindingExample in the examples/flexjs folder in the Apache FlexJS SDK directory.

1. Copy the contents of the DataBindingExample.mxml into your Application MXML file.
   Note: The current tools require that the Application name matches the Project name so you can't just copy DataBindingExample.mxml file to the source folder.
2. Copy the rest of the files from the DataBindingExample project folder to the source folder of your project.
3. Rename references to DataBindingExample in the other source files to match your Application's name.

Typical Application Project Code:

FlexJS Application Example

```
<?xml version="1.0" encoding="utf-8"?>
<js:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
                xmlns:js="library://ns.apache.org/flexjs/basic"
                xmlns:models="models.*"
                xmlns:controllers="controllers.*"
                xmlns:local="*">

    <js:initialView>
        <local:MyInitialView />
    </js:initialView>

    <js:model>
        <models:MyModel />
    </js:model>

    <js:controller>
        <controllers:MyController />
    </js:controller>

</js:Application>
```

Building

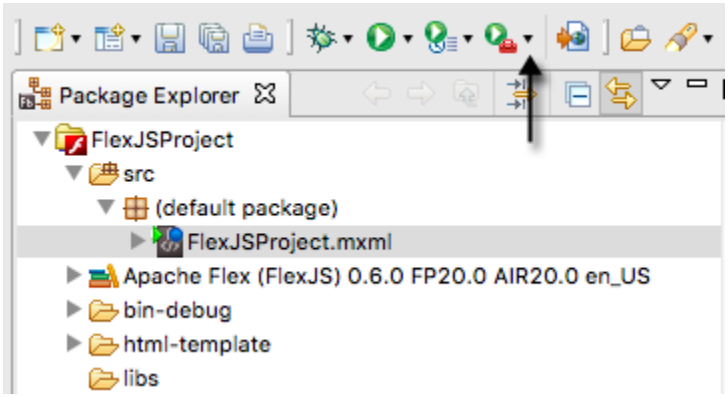
As you edit and save changes to the code, the new compiler known as FalconJX is compiling not only a SWF, but also cross-compiling your source code into JavaScript, and reporting any errors it finds. FlexJS uses a "common denominator" approach to its libraries. Where there are differences in the APIs between Flash and Javascript runtimes, the FlexJS library API will present a new common API that will work for both runtimes, but also expose the runtime-specific APIs as well. This makes it possible for the library code to be as fast as possible. Instead of calling APIs on instances of objects that wrap runtime objects, your code can directly access the runtime object itself. The danger is that if you are porting existing Flash-specific code, you won't know if you are still calling APIs that only exist on Flash. Each FlexJS library is actually two libraries, one for Flash, one for JavaScript and when your code is cross-compiled, the cross-compiler (or transpiler) will look in the JavaScript version of the library and realize that some of those Flash-specific APIs are not implemented for JavaScript.

Running/Debugging

To run the Flash version of your application, you should be able to set breakpoints and debug as you would any other Flex project.

To Debug:

1. Select the main Application file in Package Explorer

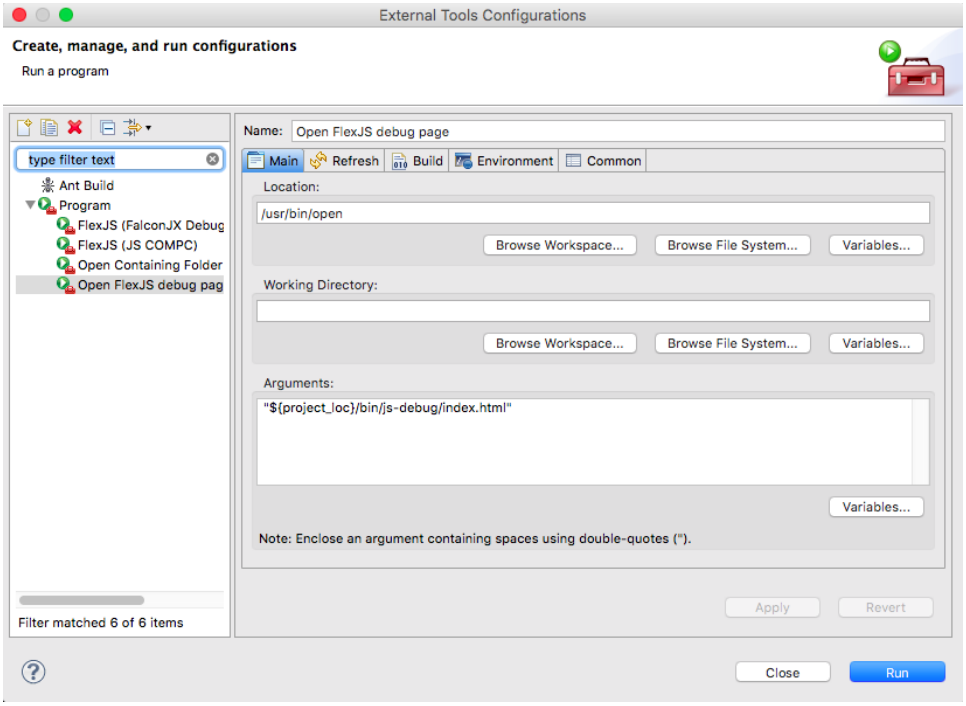


2. Click the Debug menu icon. This will open your app in the browser where you can debug and interact with the SWF version.

To Run:

1. Select the main Application file in Package Explorer
2. Click on the Run menu icon

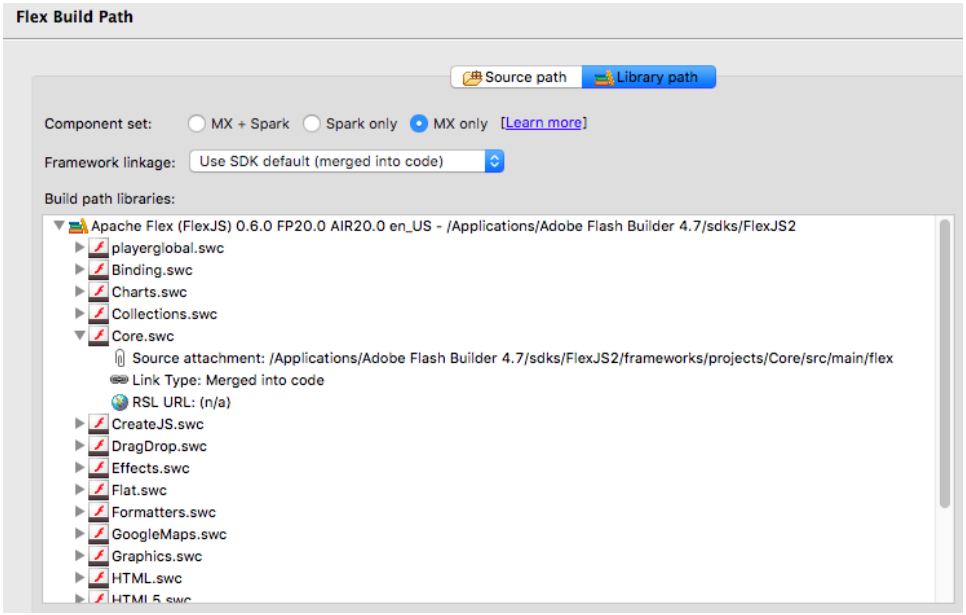
To run or debug the JavaScript version, use a browser to open the index.html in the bin/js-debug of the project, or the minified version in the bin/js-release folder. Each browser has its own JavaScript debugging tools. You can also create a External tool to open the debug index.html using the settings below (on Mac)

1. 

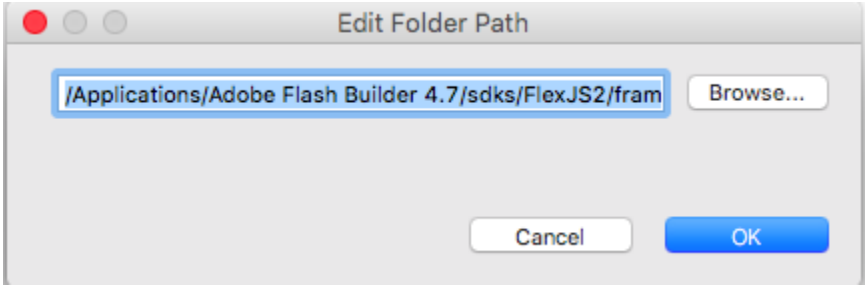
Adding Support for Opening Declarations

By default, Flex JS does not yet support opening the class declarations. You can add support manually to each FlexJS project by following the steps below:

1. Select your FlexJS project and open the Project Properties.
2. Select the Flex Build Path and then expand the Apache Flex (Flex JS) library



3. In each library you'd like to add expand and branch and double click on the source attachment icon.
4. In the dialog browse to the source path of the library and click OK. For example, "/Applications/Adobe Flash Builder 4.7/sdks/FlexJS/frameworks /projects/Core/src/main/flex" would be the path to the Core.swc source files like so:



You should now be able to open declarations of the source files.

Converting an Existing Flex Application

The FlexJS prototype is so new, that nobody has attempted to convert an existing application since significant functionality is probably missing. However, the steps should be:

1. In the Project Properties, change the Flex SDK.
2. In each MXML file, change the xmlns for mx or s (or both) to library://ns.apache.org/flexjs/basic
3. You will then have to fix lots of errors in your code where your code uses imports and API from Flex SDK classes that are not currently supported in FlexJS.

Getting Help

Please ask questions on the dev@flex.apache.org mailing list. Please start the subject of your email with "[FlexJS (legacy)]". You may file bugs at https://issues.apache.org/jira/browse/FLEX under the component "FlexJS". Remember, Apache projects like Apache Flex is mostly staffed by volunteers so response times may vary and any contributions like patches are welcome.