FlexJS: Flex for JavaScript

Presented For The Apache Foundation By
LINUX FOUNDATION

Flex is not just for making Flash apps

Presented For The Apache Foundation By
LINUX FOUNDATION

Agenda

- Who
- Why
- What
  - Highlights
  - Demo
- When
- How
  - Under the hood

Who am I?

- Alex Harui
  - Apache Flex Committer & PMC Member
  - 30+ years experience
  - Adobe Systems Inc. (almost 12 years!)
  - Flex SDK developer
  - Currently Apache Flex PMC Chair

Why FlexJS?

- Adobe Flash Player used to be in every browser
- Adobe AIR used to run on most computers
- Executives no longer carry Flash-capable devices
  - AIR apps require installation and upgrades
  - AIR apps have some fidelity issues
- Large MXML and ActionScript code bases
- Mobile needs lighter framework

What == FlexJS

- Use MXML and ActionScript to create either SWFs that run in Flash/AIR or HTML/JS/CSS files that run in browsers (or anywhere HTML/JS/CSS runs) without Flash.
  - IE8, 9, 10, Chrome, Firefox, Android, IOS
  - Mobile Apps via Apache Cordova

Several Approaches

- Emulate Flash Player
- Emulate current Apache Flex SDK
- New Framework

Emulate Flash Player

- Then you wouldn't have to rewrite any of your code.
- But that's a lot of work
- Fidelity/Performance Issues
- See JooFlash
  - http://www.jangaroo.net/applications
- Also DartFlash
  - http://www.stagexl.org

Emulate Flex SDK

- Then you'd have to change code whereever you went straight to Flash APIs
  - useHandCursor
  - blendModes, filters
- Still some fidelity/performance issues
  - Weak references
  - Dictionary
  - E4x
  - Embedded Assets
  - Class interdependencies

FlexJS: New Framework

- Designed to be cross-compiled
  - Doesn't use AS/Flash features that are hard to implement in JS
- New coding patterns support plug-ins and composition
  - Incremental feature development
  - Better performance
  - Smaller SWFs and JS downloads

Backward Compatibility

- If you have an app of 10,000 lines of MXML and 100,000 lines of ActionScript, you can rewrite it all when porting to some other JS framework, or port much less of it when using FlexJS
- Would you have re-written it anyway?
  - XML handling is cumbersome in the browser. If you were going to port your app to some other JS framework, would you have switched from XML to JSON anyway?

How much can you re-use?

- To the extent your application is MXML components glued together with ActionScript that does not access Flash APIs directly, you will be able to re-use most of your code
  - No chance if you require Flash-quality video
  - No chance right now if you use TLF
- Scan your code for “import flash.*” and “embed”
  - Gives a good first estimate
  - Events are easier to port

Familiar Constructs

- MXML DataBinding
  - Can be optimized when needed
- MXML States
- Component Names
  - Button
  - Label
  - DropDownList
  - etc

What's Different

- Skinning Model
  - Bitmap-based, at least for IE8 and non-HTML5 browsers
  - SVG-vector skinning support being investigated
    - FXG for AS, SVG for JS
- Application is not a DisplayObject
- Multiple component sets
  - Many different kinds of buttons

Demo

When

- Prototype available now
- Alpha-quality due soon

How

- New Compiler (Falcon and FalconJX)
  - MXML and AS cross-compiled to JS
  - SWC classes must have JS equivalent
  - Standard CSS copied
    - AS code must handle standard CSS
  - Custom CSS cross-compiled to JS
    - JS code queries custom CSS
  - Different HTML wrapper

Under the Hood

- FlexJS is completely new code base
  - No attempt to refactor current code base
  - No attempt to copy from current code base
  - JS implementations influence AS implementations

Browser First

- Wrap built-in HTML elements
- Encapsulate and Present
  - Identify good practices in JS world
  - Package as AS classes
- As low overhead as possible

Other Philosophies

- Parallel Frameworks
- Plug-ins
- Composition over Inheritance
- Multiple Component Sets
- Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Parallel Frameworks

- Component developer:
  - Creates both a Button.as and Button.js
  - Button.as can use Flash APIs
  - Button.js uses HTMLElements, JS and CSS
  - Higher-level components can be written in AS and cross-compiled to JS
- The application developer writes one set of code
  - Compiles and debugs in Flash
  - Cross-compiles to HTML/JS/CSS

Why SWF?

- If you can still use Flash, it should save you a whole bunch of browser-specific testing and tweaking
- Leverage existing IDEs
- ActionScript is strongly-typed so will catch bugs sooner
- ActionScript VM does runtime checking

Other Philosophies

- ✔ Parallel Frameworks
- Plug-ins
- Composition over Inheritance
- Multiple Component Sets
- Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Beads (Plug-Ins)

- Instead of one Button with every property imaginable, Button optional properties are packaged into “Beads”.
- Beads are the name for plug-ins
  - They should be highly-reusable
    - Prompt for TextArea, TextInput, ComboBox, for example.
- Component when used as sub-components don’t need as much as when used at top-level
  - Border on TextInput in ComboBox

Beads (Cont'd)

- Different code for different runtime environments
  - Toss out mouseover code on mobile, swap in touch code instead
- Use CSS to choose beads
- Wrap up a bunch of beads into a top-level component and proxy the model to the component API surface

Other Philosophies

- ✔ Parallel Frameworks
- ✔ Plug-ins
- Composition over Inheritance
- Multiple Component Sets
- Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Composition

- JIT compilers are used in the runtimes.
  - The more you re-use code, the more efficient JIT is
  - Flex startup is actually faster without JIT because it doesn't re-use as much code
- A single feature can be written once and applied in several places.
  - Text prompt example
- https://cwiki.apache.org/confluence/display/FLEX/Creating+Components

Other Philosophies

- ✔ Parallel Frameworks
- ✔ Plug-ins
- ✔ Composition over Inheritance
- Multiple Component Sets
- Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Multiple Component Sets

- A one-size-fits-all Button works in most places, but not all
- A choice of Buttons means you have more decisions to make, but you don't have to carry around excess code.
- You can use JQuery or other JS UI frameworks or a set of components that are HTML5 dependent.

Other Philosophies

- ✔ Parallel Frameworks
- ✔ Plug-ins
- ✔ Composition over Inheritance
- ✔ Multiple Component Sets
- Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Just-in-Time

- Flex initializes a bunch of managers at startup that prepare for overlapping top-level windows, tooltips, and custom cursors.
  - None of these are likely to be found in mobile apps
  - You don't even need PopUpManager unless you have floating non-modal popups
- Choose the right PopUpManager, and only instantiate it when you actually get around to showing a popup.

Other Philosophies

- ✔ Parallel Frameworks
- ✔ Plug-ins
- ✔ Composition over Inheritance
- ✔ Multiple Component Sets
- ✔ Just-in-time not Just-in-case
- Rapid prototyping is important, but end game optimization is critical

Prototyping vs Optimization

- How do you choose from so many different buttons?
  - Need utility to help you choose
  - Maybe a few heavy buttons with lots of options baked in
- More MXML tags to write.
- You will be able to toss out code you are not using.
- Debug-mode beads can give more warnings and do more parameter checking.

Help Wanted

- All kinds of contributions welcome
  - Testing
  - Development
  - Documentation
  - Examples

Summary

- FlexJS protects your MXML and ActionScript investment
- FlexJS apps will run in just about any browser
- MXML and ActionScript are enterprise-class languages
- You can directly affect the growth and development of FlexJS

Q&A

- http://flex.apache.org
- mailto:dev@flex.apache.org
  - Subject Prefix: “[FlexJS]”