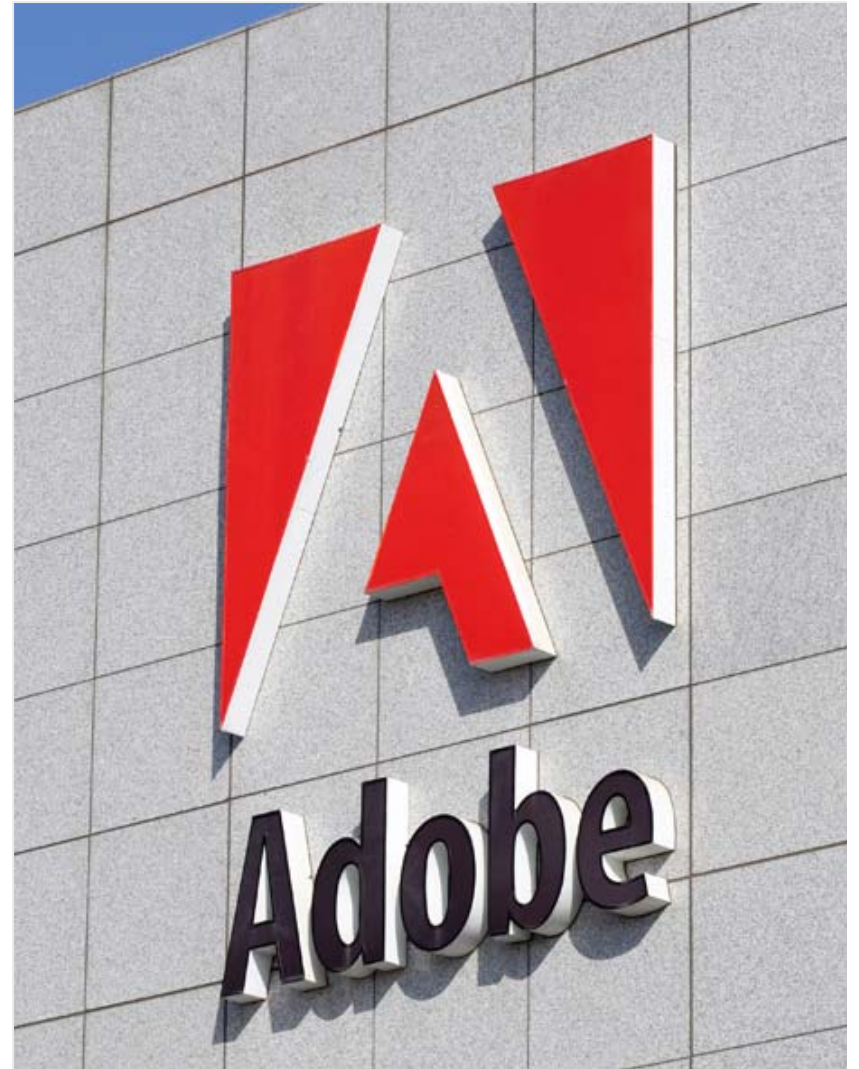


Flex Framework Internals – Part 1

Alex Harui

Adobe Systems



Topics

- More details on component lifecycle
 - Application startup
 - Component instantiation
- How to implement keyboard/focus handling in a component
- Time permitting: more info on SystemManager and Flex SWF initialization

Component Lifecycle – The Display List

- Lots of things in Flex are based on the display list you are connected to
 - Embedded fonts
 - Styles
 - Size of the Player
 - Top of the DOM
- These things are usually accessed via the `SystemManager`
- When components are created they are not connected to a display list
 - `systemManager == null`
- Your constructor therefore does not have access to these things

```
var randomWalk:UIComponent = new
    RandomWalk();

randomWalk.dataProvider = dataSet;

/* Random Walk cannot access the
    player's size, or add capture phase
    event listeners until addChild is
    called because systemManager ==
    null */

addChild(randomWalk);

/* After addChild(), systemManager !=
    null */
```

Component Lifecycle – Application

- Application is a special case.
- Also not on a display list, but
- `systemManager != null`
 - `stage == null`
 - `root == null`
- Not on a display list until after the `creationComplete` event
 - Therefore, your component isn't on a display list either
- On a display list when `applicationComplete` event is dispatched
- Purely a performance optimization
- Easy to trip over.
- Use `systemManager.stage` (but not in constructor)

Component Lifecycle – Component Instantiation

```
var myComp:UIComponent = new  
    RandomWalk()  
  
myComp.dataProvider = dataSet;  
  
...  
  
addChild(myComp);
```

Note that your child's initialize events are dispatched before yours.

```
public function RandomWalk() {  
    ...}  
  
public function set dataProvider() {...}  
  
...  
  
• Prepares set initial set of styles  
• Adds to the display list  
• Dispatches "preInitialize"  
  
protected function createChildren() {...}  
  
• Initializes accessibility, if any  
• Dispatches "initialize"
```

Component Lifecycle – Some Rules

- Don't add children in constructor
 - Won't break, but slightly less efficient
 - Best practice to override createChildren()
- Don't call getStyle() in constructor
 - Won't break, but may return incorrect information
 - Wait until createChildren()
 - Better practice to override commitProperties()
- Don't access systemManager, stage or root in constructor
 - They are null
 - Wait for createChildren() or later
- Don't access stage or root directly
 - Use systemManager instead

Component Lifecycle – Startup - Initialization

1. Application is instantiated
2. Application.systemManager is set
3. Application.dispatchEvent(new Event('preinitialize'))
4. Application.createChildren()
 1. randomWalk is instantiated
 2. The RandomWalk() constructor is called
 3. Properties are set on the component
 4. randomWalk.dispatchEvent(new Event('preinitialize'))
 5. randomWalk is added to the application (but still isn't on the display list)
 6. randomWalk.createChildren()
 7. randomWalk.dispatchEvent(new Event('initialize'))
5. Application.dispatchEvent(new Event('initialize'))

Component Lifecycle – Startup - Validation

- Phase 1
 - `randomWalk.commitProperties()`
 - `Application.commitProperties()`
- Phase 2
 - `randomWalk.measure()`
 - `Application.measure()`
- Phase 3
 - `Application.updateDisplayList()`
 - `randomWalk.updateDisplayList()`
- Phase 3
 - `randomWalk.dispatchEvent(new Event('creationComplete'))`
 - `Application.dispatchEvent(new Event('creationComplete'))`

Component Lifecycle – Startup - Completion

- Application is added to display list
- `Application.dispatchEvent(new Event('applicationComplete'))`

Flex Framework Internals – Part 1

Keyboard/Focus Handling



Keyboard/Focus Handling

- Flash Player has built-in properties, methods, and events for keyboard and focus handling. However, the Flash Player doesn't support:
 - Composition
 - Modality
- Flex overrides the most of the Flash Players keyboard/focus support.
- Flex adds new interfaces and properties for keyboard/focus support
 - IFocusManagerComponent
 - focusEnabled
- UIComponent has all of the properties of IFocusManagerComponent but does not implement it so that components can be divided between interactive components (TextInput, DataGrid) and non-interactive (Label, ProgressBar)

Keyboard/Focus Handling - Basics

- Basic Keyboard/Focus Handling is as simple as:
 - “implements IFocusManagerComponent”
 - UIComponent.keyDownHandler

```
public class RandomWalk implements
    IFocusManagerComponent
{
    override protected function
    keyDownHandler(
        event:KeyboardEvent):void
    {
        switch (event.keyCode)
        {
            case Key.DOWN:
        }
    }
}
```

Keyboard/Focus Handling - Basics

- If a component does not implement `IFocusManagerComponent`, the component focus-related properties will have no effect
 - Containers do have their `tabChildren` properties monitored
- `focusEnabled` – whether the `FocusManager` should care about other focus-related properties on this instance.
 - Used to turn off focusability in subcomponents and identify composite components
- `tabEnabled` – whether the `FocusManager` will give this component focus when the `TAB` key is pressed.
 - All parents of this component must have `tabChildren==true`.
- `mouseFocusEnabled` – whether the `FocusManager` will give this component focus when clicked by the mouse.
 - If false, the parent's `mouseFocusEnabled` is checked

Keyboard/Focus Handling - Composition

- Most components are not Composites
 - Button
 - CheckBox
 - ButtonBar!
- Composite Components have more than one internal IFocusManagerComponent or generate them at run-time, and use the Tab key to navigate between IFocusManagerComponents.
 - DataGrid/List/Tree item editors
- Some components are actually containers from the FocusManager perspective
 - Slider is a container of focusable thumbs

Keyboard/Focus Handling - Slider

- Doesn't implement IFocusManagerComponent
- Sets tabChildren = true
- Children (thumbs) are IFocusManagerComponents
- Easier to implement than composites
- We will probably re-implement as composite in future
 - Future Flex Builder tabIndex dialogs may not support this configuration

```
public class Slider
{
    public function Slider()
    {
        tabChildren = true;
    }
}

public class SliderThumb implements
IFocusManagerComponent
```

Keyboard/Focus Handling - Composition

- Implement IFocusManagerComponent
- In focusInHandler, start listening for “keyFocusChange” event
- In keyFocusChangeHandler:
 - preventDefault() so the focusManager does not process the event
 - Manage Tab keys and set focus to internal components.

```
public class DataGrid implements
    IFocusManagerComponent
{
    override protected function
    focusInHandler(event:FocusEvent):void
    {
        addEventListener(“keyFocusChange”,
        keyFocusChangeHandler);
    }

    protected function
    keyFocusChangeHandler(event:FocusE
    vent):void
    {
        event.preventDefault();
        itemEditor.setFocus();
    }
}
```

Keyboard/Focus Handling - Summary

- Almost all Flex components were implemented w/o compositing from a FocusManager perspective
- Exceptions:
 - Slider
 - List/Tree
 - DataGrid
- Most of your components probably won't need to deal with compositing either.

SystemManager

- Root of the SWF
- First class that is instantiated
- Controls and coordinates the initialization of the application.
 - Instantiates and displays Preloader
 - Instantiates Application
- Manages layers of children for popups, cursors and tooltips
- Helps manage classes in ApplicationDomains

Flex SWF Initialization

- Flex SWFs are two frame movies
- The first frame contains a `SystemManager`, the `Preloader`, the `DownloadProgressbar` and a few helper classes
- The second frame contains the rest of the framework, the application code and assets (embedded fonts, images, etc)
 - Your custom component and all of its assets are in the second frame

Flex SWF Initialization (Frame 1)

- Enough bytes for frame 1 are streamed in
- Flash Player executes those bytes
 - Instantiates a SystemManager
 - SystemManager tells player to stop at the end of the frame
 - SystemManager creates a Preloader
 - Preloader creates a DownloadProgressBar
 - Preloader and DownloadProgressBar watch rest of bytes streaming in.
 - SystemManager listens for frameEnd event.
- Flash Player stops at end of frame and continues to stream in SWF

Flex SWF Initialization (Frame 2)

- Once all bytes are in, Flash Player sends a frameEnd event.
 - SystemManager instantiates the Application
 - Sets Application.systemManager to itself
 - SystemManager listens for “preloaderDone” event
 - Application creates its children
- Flash Player sends enterFrame and render Events
 - LayoutManager validates the created children
- Eventually, the application dispatches its creationComplete event
 - Preloader begins removal of DownloadProgressBar
 - Preloader send “preloaderDone” event
 - SystemManagers adds Application to display list.
 - SystemManager tells Application to dispatch “applicationComplete” event