More Flex

CS174

Chris Pollett

Dec. 1, 2008.

Outline

- CSS
- Actionscript
- Event Handling
- Working with Media

CSS

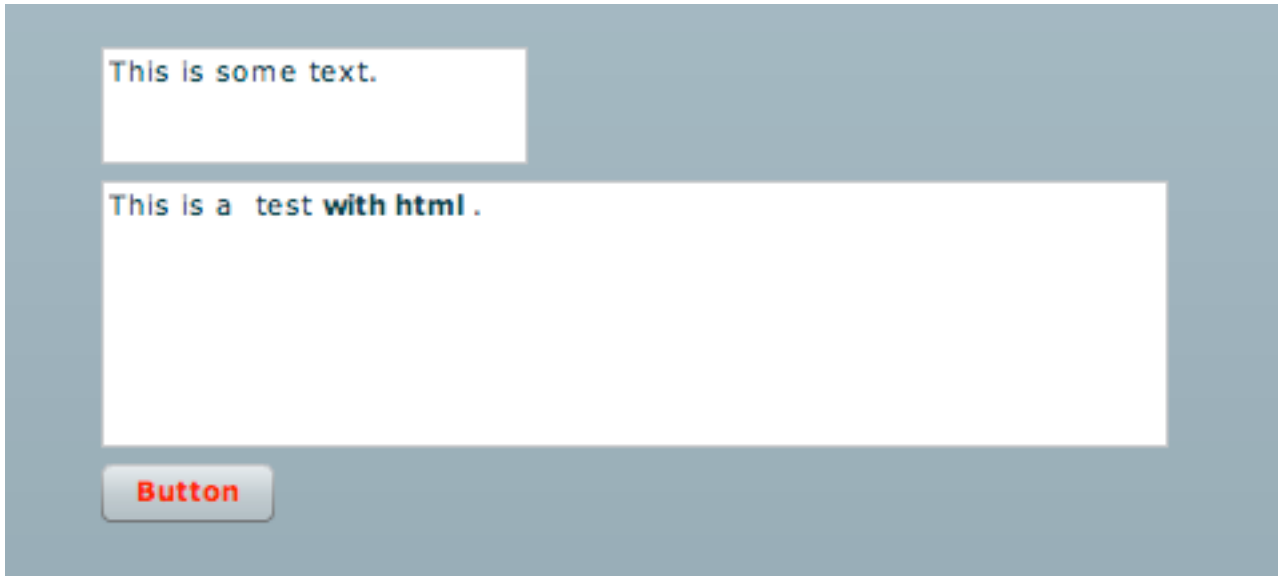
- Styles can be used in mxml in a similar fashion to HTML.
- The style tag is <mx:Style> (potentially with a different prefix) not <style> however.
- External styles can be includes using <mx:Style src="stylesheet.css" />
- To set the style of an element one sets its styleName attribute.

Example with CSS and also of TextArea and comments

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"  width="100%"
     height="100%" >

<!-- styles can be used with Flex (notice html style comment) -->
<mx:Style>
  /*     C style comment   */
  .red
  {
     color: red;
  }
</mx:Style>
   <mx:VBox>
      <mx:TextArea> <mx:text>This is some text.</mx:text></mx:TextArea>
      <mx:TextArea width="400" height="100">
         <mx:htmlText><![CDATA[This is a  test <b>with html </b>.]]></mx:htmlText>
      </mx:TextArea>
      <mx:Button label="Button" styleName="red" />
   </mx:VBox>
</mx:Application>
```

What Example Looks Like



Actionscript

- Actionscript is a supped up version of Javascript.
- It can appear inline as when we did things like last day:

```
<mx:Text id="out" text="{'the output is:' + someid.text}"
    />
```

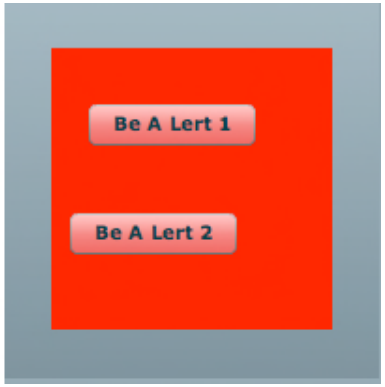
- It can be included in your mxml document between <mx:Script> </mx:Script> tags
- External scripts can also use be used by doing:

```
<mx:Script src="my_script.as" />
```

Simple Actionscript Example

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"  width="200" height="200" >
<mx:Script>
function imclicked()
{
 var a = 30;
 mx.controls.Alert.show('The world needs more lerts '+a);
 //since this function does not specify types and say what package it is in, it will generate warnings
}
</mx:Script>
<mx:Style>
  Canvas { background-color:red; }
</mx:Style>
<mx:Canvas width="150" height="150"> <!--Notice how you can use a Canvas layout for positioning -->
  <mx:Button id="bealert1"  x="20" y="30"
     label="Be A Lert 1" click="mx.controls.Alert.show('The world needs more lerts'); " />

  <mx:Button id="bealert2"  left="10" bottom="40"><!--right and bottom based on parent container -->
      <mx:label>Be A Lert 2</mx:label>
       <mx:click>imclicked();
       </mx:click>
   </mx:Button>
</mx:Canvas>
</mx:Application>
```



What's new in Actionscript versus Javascript

- Actionscript supports packages and classes.
- Packages are used to give a namespace for a collection of classes.
- General format of a class definition looks like:

```
package my_package
{
   import some_package; //a list of packages
   public class MyClass extends Something
   {
     // some var declarations
     public function MyClass(){} //constructor
     /* other methods which could be static,
          public, private, protected (available to subclass), internal (available within package)*/
   }
}
```

- The Alert on the previous slide shows how we could invoke a method once it is defined.
- In Actionscript, variables can be declared either with or without a type:

  var my_variable:Number;

- Some example types: String, Number, int, uint, Boolean, Date, Array, XML.
- Function and method declaration can also specify input and return types:

  internal function my_method(my_arg:int) : uint {/* body */}

More on Actionscript

- In addition to classes, like Java, Actionscript also allows you to define interfaces.
- Actionscript does not allow two methods with the same name but different signatures (overloading).
- We've already seen you can extend existing classes like in Java.
- The control structures, expressions, Array's, Object's, try-catch, are otherwise like in Javascript.
- As with the Javascript DOM, each tag in MXML corresponds to a Actionscript object and so you can use Actionscript to set properties of a tag.

Event Handling

- Events can be handled by either adding a handling like we did when we did the alert example a couple of slides back or by using something akin to the DOM 2 event model we discussed with Javascript:

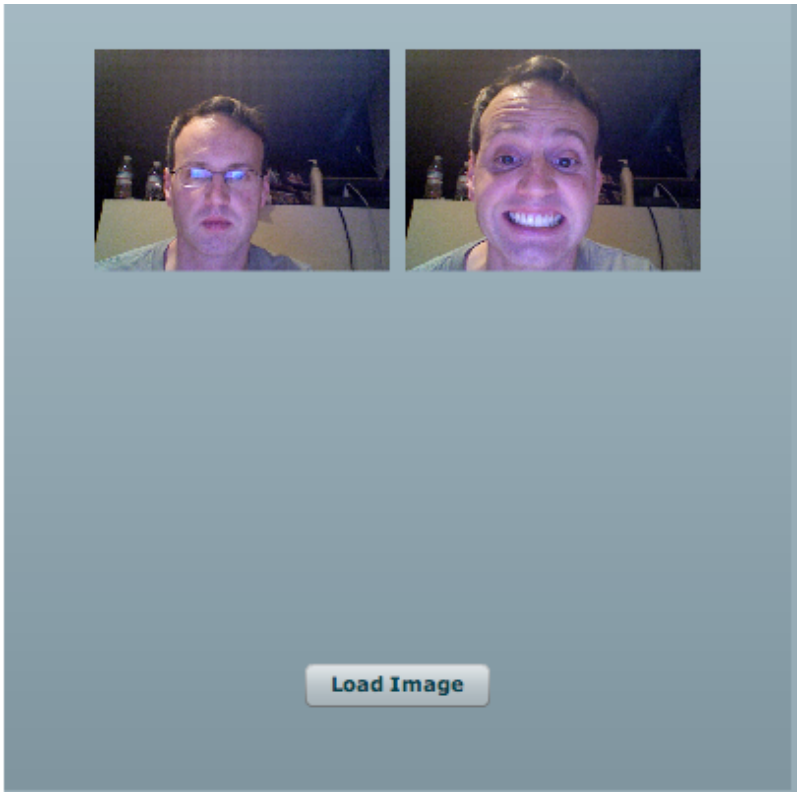
```
button.addEventListener(MouseEvent.CLICK,
    clickHandler)
function clickHandler(evt:MouseEvent):void {/*code */}
button.removeEventListener(MouseEvent.CLICK
    clickHandler);
```

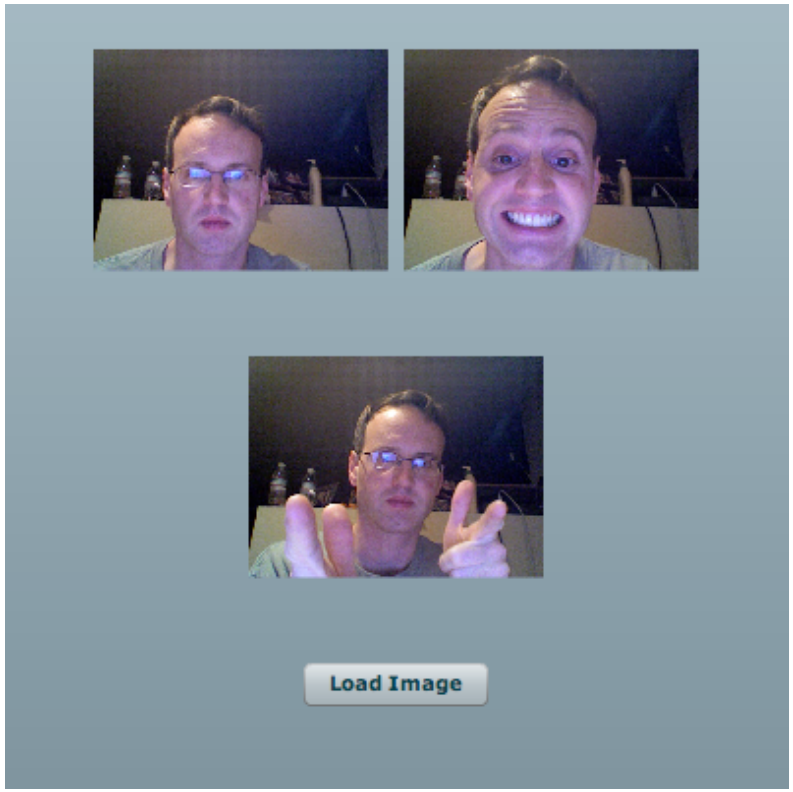
Working with Media -- Images

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"  width="400" height="400"
      initialize="init()" >
   <mx:Script>
    <![CDATA[
      [Embed(source="me3.jpg")] /* this add me3.jpg directly into the .flv file that will be produced
                                          other two images below come from external files */
      private var meAsset:Class; //will represent image file above
      private function init():void { me3.source = meAsset; }
      ]]>
   </mx:Script>
   <mx:HBox>
      <mx:Image source="me1.jpg" width="150" height="150"/>
      <!-- need to use webserver not local filesystem -->
      <mx:Image id="me3" width="150" height="150"/>
    </mx:HBox>
    <mx:Image id="image2" autoLoad="false" width="150" height="150" />
    <mx:Button label="Load Image" click="image2.load('me2.jpg');" />
</mx:Application>
```

The Embed directive can also be used to create skins for Buttons, etc. You can set style properties: down-skin, up-skin, and over-skin. You can set the icon attribute of the button as in icon="@Embed(my.gif)".

Image Example Before and After Click





Video

```
<?xml version="1.0" encoding="utf-8" ?>
  <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" width="400"
    height="400" >
    <mx:Panel title="My Video" >
      <mx:VideoDisplay id="myViewer" source="fred_ott.flv"
          volume="{volumeControl.value}" />
      <!-- note the source is an http source; trouble if play on filesystem -->
      <!-- you should look up Fred Ott -->
      <!-- you can create .flv files using ffmeg (Mac, Linux, Windows).
           Or google for your favorite other tool. -->
      <mx:Label text="{myViewer.playheadTime.toPrecision(2)}" />

      <mx:ControlBar>
        <mx:Button label="Play" click="myViewer.play();" />
        <mx:Button label="Pause" click="myViewer.pause();" />
        <mx:Button label="Stop" click="myViewer.stop();" />
        <mx:HSlider id="volumeControl" maximum="1" width="80" />
      </mx:ControlBar>
    </mx:Panel>
</mx:Application>
```

SoundEffect and Sound can be used to play sound.

What Video Example Looks Like