ACTIONSCRIPT SESSION 4

Roy McElmurry

For Loops

Syntax:

```
for (var i:int = 0; i < condition; i ++) {
    //normal for loop
}

for (var name:type in array) {
    //loops over the indices of the array
}

for each (var name:type in array) {
    //loops over the values of the array
}
```

More: http://www.darronschall.com/weblog/2005/12/actionscript-3s-new-loop.cfm

TextFormat

Syntax:

```
import flash.text.*;
var textField:TextField = new TextField();
…
var format:TextFormat = new TextFormat(font, size, color, …);
textField.setTextFormat(format);
```

- The *TextFormat* object can be used to set many other properties as well.

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/text/TextFormat.html

TextFieldAutoSize

Syntax:

```
import flash.text.*;
var textField:TextField = new TextField();
…
var format:TextFormat = new TextFormat(font, size, color, …);
textField.autoSize = TextFieldAutoSize.LEFT;
textField.setTextFormat(format);
```

- The auto size property must be set so that the *TextField* knows how it should respond to formatting
- In this case in treats the text as left justified

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/text/TextFieldAutoSize.html

FlashVars in HTML

Syntax:

```
<div>
    <object type="application/x-shockwave-flash" data="yourfile.swf"
                width="550" height="400" >
        <param name="yourfile" value="yourfile.swf" />
        <param name="FlashVars" value="key=value&key=value…" />
    </object>
</div>
```

- FlashVars are a way for us to load a flash movie with parameters
- The format is the same as that for HTML query strings
- You can also just add a query string to the end of the data attribute

FlashVars in AS

Syntax:

```
import flash.display.LoaderInfo;
…
var params:Object = LoaderInfo(this.root.loaderInfo).parameters;
for (var key:String in params) {
        //do stuff with the keys of the FlashVars
        //value = params[key]
}
```

- You can treat the params object as if it were an associative *Array*

ExternalInterface

Syntax:

```
import flash.external.ExternalInterface;
…
ExternalInterface.call("jsFunction", param1, param2,…);
…
var returnValue:type = ExternalInterface.call("jsFunction", param1,…);
```

- ExternalInterface allows us to talk with JavaScript
- You can even get a return value from the javascript function, and it can be of any Actionscript type

ExternalInterface

Syntax:

ActionScript

```
import flash.external.ExternalInterface;
…
ExternalInterface.addCallback("name", myFunction);
…
public function myFunction(param:type, …):type {/*do stuff*/}
```

Javascript

```
$("flashMovie"). name(parameters);
```

- The id flashMovie must be attached to the object tag that you use to embed the flash movie

Security Issues

- Now that we have allowed our flash movie to talk to Javascript and vice versa some issues arise
- So now we can call any functions that the page has
- What if we make anonymous functions?
- Lucky (or disappointingly) Javascript cannot do things like read files or connect to databases, but we can still cause some damage

ExternalInterface Security

Syntax:

HTML

```
<object type="application/x-shockwave-flash" data="yourfile.swf"
        width="550" height="400" >
    <param name="yourfile" value="yourfile.swf" />
    <param name="allowScriptAccess" value="always" />
</object>
```

Actionscript

```
import flash.system.Security;
…
Security.allowDomain("*");
```

- "*" allows all domains, you can specify a specific one instead if you are concerned with security

Code Interactions

- FlashVars
  - Allows HTML to talk to flash
  - Allows PHP to talk to flash
- ExternalInterface
  - Allows Javascript to talk to flash
  - Allows flash to talk to Javascript

XML

Syntax:

```
import flash.net.*;
…
public function loadXML():void {
        var xmlLoader:URLLoader = new URLLoader();
        xmlLoader.addEventListener(Event.COMPLETE, readXML);
        xmlLoader.load(new URLRequest("file.xml"));
}
public function readXML(e:Event):void {
        var xml:XML = new XML(e.target.data);
        //do stuff with the XML
}
```

- Actionscript can load and process XML data

More Security Issues

- Flash allows you to process files either locally or remotely, but not both
- This guards against stealing data
- The default is to be able to read remote files
- There are several ways to fix this issue in decreasing convenience order
  - Visit this site and add whatever directories you want to the trusted directories list for your flash environment
  - Load your files onto webster and test there
  - Add the -use-network=false option when using mxmlc

More: http://labs.mstudio.com/?p=31

XML vs XMLList

- When handling XML data you can use the XML object and the XMLList object
- These objects behave similarly and in most cases you think of them as being the same thing
- You can even think of an XMLList as simply a collection of XML objects
- When you create an XML object, that object represents the root node of the XML

Traversing XML

Syntax:

```
var xml:XML = new XML(e.target.data);
…
//access each node in the xml with name "nodeName"
for each (var node:XML in xml.nodeName) {
    //access the "attrName" attribute of the node
    var attr:String = node.@attrName;
    //get the text in the node
    var value:String = node.text();
}
```

- Traversing an XML object is much simpler in Actionscript

MXML

- MXML is a type of XML
- MXML describes the layout of user interface components in a flash application
- Instead of tedious addChild calls and setting x, y, width and height values, we can describe the layout relationships just like we would in html
- You can attach external Actionscript files
- You can attach external stylesheets that are very closely in line with CSS
- MXML is not a separate language, it is shortage for regular ActionScript and is directly translated to it

MXML Syntax

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns=http://www.adobe.com/2006/mxml
    layout="absolute" height="400" width="400">

    <!– Your MXML goes here ->

</Application>
```

- MXML looks a lot like HTML

UI Components

Syntax:

<Text text="**blah**" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/Text.html

Syntax:

<TextInput text="**blah**" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/TextInput.html

Syntax:

<Button label="**blah**" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/Button.html

UI Components

Syntax:

<Alert text="**blah**" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/Alert.html

Syntax:

<RadioButton groupName="**blah**" label="**blah**" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/RadioButton.html

Syntax:

<Image source="@Embed('**location**')" />

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/Image.html

More: http://livedocs.adobe.com/flex/3/langref/mx/controls/package-detail.html

Styles

Terrible Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns=http://www.adobe.com/2006/mxml
    layout="absolute" height="400” width="400”>

    <Text fontFamily=“Garamond” fontSize=“12pt” text=“blah” />

</Application>
```

Styles

Bad Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns=http://www.adobe.com/2006/mxml
      layout="absolute" height="400" width="400">
      <Style>
            @namespace "library://ns.adobe.com/flex/mx";
            Text {
                  fontFamily: "Garamond",
                  fontSize: 12pt;
            }
      </Style>

      <Text text="blah" />
</Application>
```

Styles

Good Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns=http://www.adobe.com/2006/mxml
      layout="absolute" height="400" width="400">
      <Style source="ex.css" />
      <Text text="blah" />
</Application>
```

ex.css

```
@namespace "library://ns.adobe.com/flex/mx";

Text {
      fontFamily: "Garamond";
      fontSize: 12pt;
}
```

More: http://opensource.adobe.com/wiki/display/flexsdk/CSS+Namespaces+Support

The Bad Part of Styles

- Flex makes a distinction between style properties and attributes of UI components
- This means that some attributes, most notably, x, y, width and height cannot be set in your css file
- Instead you can use the verticalCenter and horizontalCenter style properties
- Instead of giving a UI components css class you give it a styleName
- You need that icky line at the top to prevent warning messages

ActionScript

- In the same way that we attach JavaScript to HTML, we attach ActionScript to MXML
- We could write code obtrusively in the MXML file, but this would be bad style
- Many Javascript/HTML paradigms are also present with Actionscript/MXML

ActionScript

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns=http://www.adobe.com/2006/mxml
    layout="absolute" height="400" width="400">

    <Style source="ex.css" />

    <Script source="ex.as" />

    <Text text="blah" />
</Application>
```

Containers

Syntax:

<VBox>…</VBox>
<HBox>…</HBox>

More: http://www.adobe.com/livedocs/flex/2/langref/mx/containers/Box.html

Syntax:

<Canvas>…</Canvas>

More: http://www.adobe.com/livedocs/flex/2/langref/mx/containers/Canvas.html

Syntax:

<Panel title="**blah**">…</Panel>

More: http://www.adobe.com/livedocs/flex/2/langref/mx/containers/Panel.html

More: http://www.adobe.com/livedocs/flex/2/langref/mx/core/Container.html

In Sum

- MXML in junction with ActionScript and CSS is a browser independent alternative to traditional web pages
- Another cool benefit is that these apps can be run on the desktop if you install the Adobe Air Player
- Flash is really awesome!