ACTIONSCRIPT SESSION 1

Roy McElmurry

Terminology

ActionScript: an object-oriented programming language similar to JavaScript

MXML: A flavor of XML that helps simplify user interface construction code

Flash: An authoring program that assists the user in creating graphical ActionScript code

Flex: An interactive development environment and a software development package that extends ActionScript with MXML

More: http://en.wikipedia.org/wiki/ActionScript

Actionscript 3.0

- An object oriented scripting language
- Very similar to JavaScript, reminiscent of Java
- The Adobe Flash program writes this code for you
- A .swf file is the compiled version of Actionscript

Why Flash

What's cool about Flash:

- It is installed on almost every computer
- Appears the same on all machines
- We can write games and small programs easily
- Flash allows us to put video on the web easily
- Flash can be run on the desktop, outside of the browser

Getting Started

Visit the following webpage and follow the instructions

- *Coming Soon*

What you need to do

- Download the Flex SDK
- Install the StandAlone Flash player
- Start writing some ActionScript Files

Variables

Syntax:

var **name**:**type** = **value**;

- There are many types of variables one can use
  - int, Number, String, Boolean, Object
- The value you give you variable must be consistent with the type you specified
- The name of your variable can be anything that starts with a letter or underscore, with a few exceptions
- Specifying type is not mandatory, but is good practice

Comments

Syntax:

```
//One Line
/*
    Multiline
*/
```

- Comments are very important: leave them for yourself and for others
- Don't have unnecessary comments clutter your code
- You can also use them to temporarily "turn off" some code

Arrays

Syntax:

var **name**:Array = [**value1, value2,…,valuen**];
var **name**:Array = new Array(**value1, value2, …, valuen**);

- Arrays are used to store multiple values that we know are related or have meaning together
- Arrays are also useful in junction with *for* loops

Accessing Array

Syntax:

name[**index**]
name[**index**] = **value**;

- The brackets allow you to grab just one of the values stored in an array
- Indexing starts at zero, so the first value in the array is at index zero
- The first piece of syntax simply access the index
- The second piece of syntax is for reassigning the value of the array at the given index

If Statements

Syntax:

```
if (condition) {
    //statements
}
```

- If statements are used to conditionally run some code
- These can be done in combination with *else if* and *else* statements to run different code based on mutually exclusive tests

If/Else if/Else

Syntax:

```
if (condition) {
    //statements
} else if (condition) {
    //statements
} else {
    //statements
}
```

- You can have as many *else if* statements as you want
- You may or may not include the *else* case

For Loop

Syntax:

```
for (declaration; condition; update) {
    //Statements
}
```

- *for* loops are used to do similar code a definite number of times
- Typically we define a new loop variable and have our condition and update based on it

Typical For Loop

Syntax:

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

- Notice that we have a variable *i* which we could use during the statements section, this is common practice
- Typically the loop variable is called *i, j, or k*

While Loop

Syntax:

```
while (condition) {
    //statements
}
```

- *while* loops are used for executing similar code when we don’t know how many times we will do it
- For instance we may want to grab user input until they give us a certain piece of information

Functions

Syntax:

```
function name(parameter1:type,…,parametern:type):type {
    //statements
}
```

- A function is used to capture a procedure that you may want to use several times
- With functions you can reuse code and make it much easier to read

Using functions

Syntax:

name(**value1**, …, **valuen**);
var **name**:**type** = **functionName**(**value1**, …, **valuen**);

- To use a function you simply call its function name, and give it whatever parameters it needs
- If the function has a return type other than void, you can capture that returned value by assigning the function call to a variable

Outside functions

- There are many functions you may want to use that you didn't write yourself
- To do so, you must use the dot notation, such as *Math.sqrt(4);*

Math Function	
Math.sqrt(**number**);	Square Root
Math.abs(**number**);	Absolute Value
Math.max(**number1**, **number2**);	Maximum, there is also a min
Math.round(**number**);	Round
Math.E, Math.PI	E and Pi values

Actionscript Files

Template:

```
package  {
	import flash.display.Sprite;

	[SWF(backgroundColor="#ffffff", frameRate="24", width="550", height="400")]
	public class NAME extends Sprite {
		//STUFF HERE
	}
}
```

- *Actionscript files are saved with an "as" extension as in "HelloWorld.as"*

Text Fields

Example:

```
import flash.text.*;
…
var myTextField:TextField = new TextField();
myTextField.text = "Hello World";
myTextField.x = 100;
myTextField.y = 100;
```

- *There are three kinds of text fields static, dynamic and input*

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

Text Fields Types

Example:

```
var myTextField:TextField = new TextField();
myTextField.text = “Hello World”;

myTextField.type = TextFieldType.INPUT;
myTextField.border = true;
```

- *You choose to have a dynamic or input text box by changing the type field of your TextField variable*

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

Shapes

Example:

```
import flash.display.*;
…
var myShape:Shape = new Shape();
myShape.graphics.beginFill(0x000000);
myShape.drawCircle(100, 100, 50);
```

- *The graphics object has many kinds of shapes we can draw*

More: http://www.adobe.com/livedocs/flex/2/langref/flash/display/Shape.html

Graphics Functions

- A Lot of the graphics methods are just like with the graphics object that we saw in the DrawingPanel

Graphics Function	
beginFill(color:uint)	Sets the "paintbrush" color
drawCircle(x:Number, y:Number, radius:Number)	Draws a circle
drawRect(x:Number, y:Number, width:Number, height:Number)	Draws a Rectangle
moveTo(x:Number, y:Number)	Moves the "paintbrush" to (x,y)
lineTo(x:Number, y:Number)	Draws line to (x,y)
clear()	Erases everything

More: http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/display/Graphics.html

Making Them Appear

Syntax:

addChild(**SOMETHING**);

- *Objects are not by default displayed on the screen, instead we must add them to the screen by calling addChild()*
- *You can actually add things as the children of objects other than the screen*

More: http://www.flashandmath.com/intermediate/children/index.html

Example - Shapes

```
package  {
        import flash.display.Sprite;
        import flash.display.*;

        [SWF(backgroundColor="#ffffff", frameRate="24", width="550", height="400")]
        public class Shapes extends Sprite {
                public function Shapes():void {
                        var myShape:Shape = new Shape();
                        myShape.graphics.beginFill(0x000000);                //black
                        myShape.graphics.drawCircle(50, 75, 25);
                        myShape.graphics.drawRect(150, 150, 100, 75);

                        addChild(myShape);
                }
        }
}
```

Putting your swf on the web

Syntax:

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

- *There are many other parameters you can pass to your object tag with an inner param tag*
- *IE has problems with the object tag, click the more link for tips on how to embed flash in IE*

More: http://kb2.adobe.com/cps/127/tn_12701.html

More: http://www.w3schools.com/flash/flash_inhtml.asp

Shape Properties

Shape Properties	
shape.x	X coordinate w.r.t. its parent
shape.y	Y coordiante w.r.t. its parent
shape.rotation	Rotation w.r.t. its parent
shape.width	Width w.r.t. its parent
shape.height	Height w.r.t. its parent

- *Shapes have many properties including the above*
- *Each of the above is with respect to the shapes parent*

More: http://www.adobe.com/livedocs/flex/2/langref/flash/display/Shape.html

Adding Children To Sprites

Syntax:

sprite.addChild(**SOMETHING**);

- *So far we only know of Shape and TextField objects, but actually we also know about the Sprite class which we extend.*
- *We can instantiate more sprites and add children to them to create more complicated designs*

More: http://www.flashandmath.com/intermediate/children/index.html