Building Well Architected Applications

In this unit, you will learn how to architect a Flash Lite application.

Objectives

After completing this unit, you should be able to:

- ►Execute ActionScript 2 code on a keyframe
- ►Reference the capabilities of the device
- ►Declare variables, their datatype and assign default values
- ►Understand, set and override default properties for visual objects
- ►Create user defined functions
- ►Interact with ActionScript 2 text fields

Data Typing Objects

ActionScript 2 allows you to specify, in advance, what type of data a variable can contain: its data type.

Benefits of data type declaration:

- ▸A best practice because it can help prevent some programming errors.
- ▸The Actions panel then provides code-hinting for the typed identifier.

Data types are classes

All data types in ActionScript are class names, discussed later in this course. For now, this just means your data types:

- ▶Always begin with a capital letter.
- ▶Will be a noun describing a category of possible objects.

A data type is specified immediately after a newly declared identifier (whether it's a variable name, or the instance name of an object), and is separated from the variable name by a colon (:). For example:

```
var variableName:DataType = default value;
```

There are three core data types: `String`, `Number`, and `Boolean`.

String data type

- String data is a list of characters - any characters.
- You may have a `String` that is a numeric value, but ActionScript will still treat it as a string of characters that happen to be wholly or partly numeric, which is not the same thing.

The following examples show `String` values.

```
var firstName:String = "Fred";
var tagLine:String = "Silly Soda";
var formattedPrice:String = "$7.95";
```

- Values assigned to `String` variables are surrounded by quotation marks ("). You can use double quotes (" ") or single quotes (' '), so long as you match them.

Number data type

- ▶ If you can perform mathematic computations with it, it belongs to the `Number` data type.
- ▶ ActionScript 2.0 doesn't distinguish between floating point numbers (with decimal points) and other numbers.

The following examples show `Number` values.

```
var playerScore:Number = 1000;
var itemPrice:Number = 5.95;
var xPosition:Number = 272.3;
```

Adobe

Boolean data type

- ▶Boolean values are `true` or `false`, which are keyword literals in ActionScript 2.0.
- ▶A variable declared to hold the `Boolean` data type can only have the value `true` or `false` assigned to it.

```
var adminStatus:Boolean = true;
var visible:Boolean = false;
```

- ▶`True` and `false` values do not have quotes -they are built into the language.

Understanding data type conversion

- ▸If you try to put the wrong type of data into a variable, the Flash Lite Player may throw a "data type mismatch" error.
- ▸You need to convert your data to the correct data type (if possible). ActionScript 2.0 supports conversion functions for this, including:
  - `String(value)`: converts a `Number` value to a `String`
  - `Number(value)`: converts a `String` value to a `Number`, or `NaN` ("not a number")

Practically, you're only likely to run into this issue in two situations:

- ▸Retrieving from the text property of a TextField
- ▸Assigning a value to the text property of a TextField

Retrieving from the text property of a TextField

- ▶Anything you retrieve from the text property of a `TextField` will have the `String` data type, even if they're numeric characters.
- ▶ You need to convert them to a `Number` before doing any math.

```
var txtInput1:TextField;
var txtInput2:TextField;
var total:Number;

// convert input to Number before adding
// else you'll get concatenation
total = Number(txtInput1.text) + Number(txtInput2.text);
```

Assigning a value to the text property of a TextField

- ▶Any value assigned to the text property of a `TextField` for display must be the `String` data type.
- ▶If you do math and want to display the result, you must convert the result to a `String` before assigning it for display.

```
var txtInput1:TextField;
var txtInput2:TextField;
var txtResult:TextField;
var total:Number;

// convert text input to Number before adding
// otherwise you'll get concatenation
total = Number(txtInput1.text) + Number(txtInput2.text);

// convert numeric result back to String
// else you'll get a data type mismatch error
txtResult.text = String(total);
```

Using the trace() Function

- ▸The Flash Player includes a number of built-in global functions you can use anywhere in your code -including one called `trace()`. The `trace()` function allows you to check what is inside of a variable.
- ▸You type your variable name inside of parentheses next to the name of the function. This is known as passing an argument into the function. (Another word for argument is parameter.)
- ▸You use the `trace()` function in this way:

```
var userName:String = "Fred";
trace(userName);
```

In this particular case, the argument is your variable. In response, when you test your movie, the Debug Flash Player used inside the Flash 8 authoring tool will evaluate the variable and display the value it contains in the Output panel.

Using System.capabilities

- To provide appropriate content to as many users as possible, you can use the `System.capabilities` object to determine the type of device a user has.
- The properties of the `capabilities` object in the `System` class identify the abilities of the particular system and player hosting a SWF file.
  - You can then either specify to the server (assuming a data plan) to send different SWF files based on the device capabilities
  - You can tell the SWF file to alter its presentation based on the capabilities of the device.

Using System.capabilities.hasMP3

`System.capabilities.hasMP3` specifies if the device has a MP3 decoder. This value is:

- ► `true` if the player is running on a system that has an MP3 decoder;
- ► `false` otherwise.

Using System.capabilities.has4WayKeyAS

`System.capabilities.has4WayKeyAS` is a read-only `Boolean` value that is

- `true` if the player can execute ActionScript attached to `keyEvent` handlers associated with the right, left, up and down keys;
- `false` otherwise.

If the value of this variable is `true`, when one of the four-way keys is pressed, the player first looks for a handler for that key.

- If none is found, Flash control navigation is performed.
- However, if an event handler is found, no navigation action occurs for that key.

Walkthrough 1: Declaring Objects and Data Typing

In this walkthrough, you will perform the following tasks:

- ▸Declare instance names for each object used in the application
- ▸Data type each object
- ▸Convert an objects' data type

Controlling Application Flow

One way to efficiently reuse code is to write and reference functions. Examples of repeatedly executed actions:

- ▶Loading a SWF
- ▶Calculating a shopping cart total
- ▶Tracking a user's position in the application

Functions provide a way to quickly execute repeated actions without needing to rewrite blocks of code.

Calling functions and passing arguments

- ▸To call - or "invoke" - a function in your code, you write the name of the function followed by parentheses.
- ▸If you must pass information to the function for it to do its work, you refer to each piece of information in a comma-separated list inside these parentheses. .
- ▸Often the information passed will be in a variable (or instance name, or property). Other times, you may pass literal `String`, `Number`, or `Boolean` values.

You always use parentheses when calling a function, whether or not you're passing any arguments to it.

```
functionName([argument1, argument2, …]);
```

Function examples:

```
// getTimer() takes no arguments
var elapsedMilliseconds:Number = getTimer();

// trace() takes one argument
// here a variable containing the argument value is passed
trace(elapsedMilliseconds);


// getURL(): one required, one optional arg
// here two literal String values are passed
getURL("http://www.adobe.com", "_blank");

// one may be called as an argument to another
// nested function result is passed to the outer
trace(getTimer());
```

Writing a user defined function

The general syntax for declaring a user defined function is:

```
function doThis([arg1:DataType,…]):ReturnType
{
 var localVariable:String;
 // do something
 return a value;
}
```

Returning or not returning data from a function

You call or invoke functions in your code to calculate values or cause a particular thing to happen. Generally speaking, functions come in two flavors.

- ▸Calculate something, and return a result to wherever they were called in your code.
- ▸Do something in the Flash Player, but do not return any result.

If a function returns a value, its return type must match the type of data returned.

```
function add(n1:Number, n2:Number):Number
{
 return n1+n2;
}
```

Returning or not returning data from a function

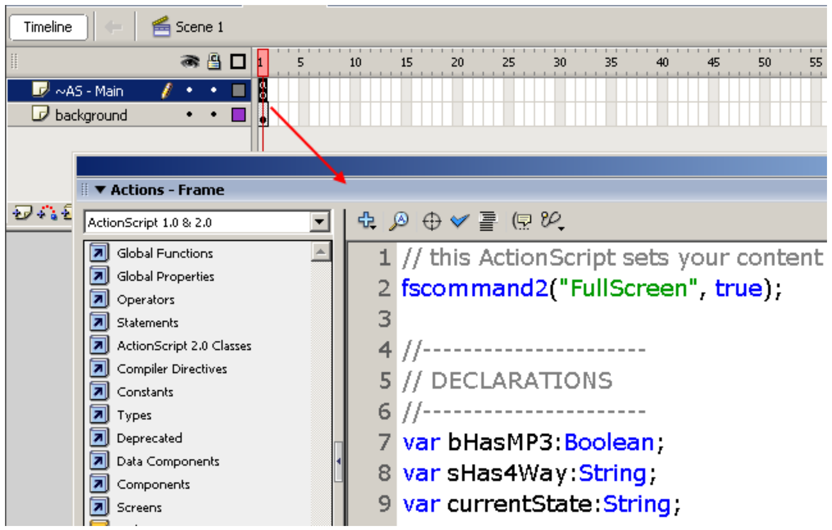
- If a function does something, but returns no value, its return type is `Void`.

```
function toggle(clip:MovieClip):Void
{
 if (clip._visible == true)
 {
  clip._visible = false;
 }
 else
 {
  clip._visible = true;
 }
}
```

Understanding function placement

The Flash IDE allows you to attach code in these general locations:

- ▶ Keyframes within the FLA
- ▶ External script files (`#include "myFile.as"`)
- ▶ External Class files (discussed at length, later in this course)
- ▶ "on" `MovieClip` symbol instances within the FLA
- ▶ "on" `Button` symbol instances within the FLA



```
// this ActionScript sets your content
fscommand2("FullScreen", true);

//---------------------
// DECLARATIONS
//---------------------
var bHasMP3:Boolean;
var sHas4Way:String;
var currentState:String;
```

Actions attached to a keyframe

Understanding function placement

- ▸The best practice is to place all ActionScript on keyframes, or in external script or class files.
  - ▪While it is possible to place ActionScript on `MovieClip` and `Button` symbol instances, this is now a discouraged practice because it quickly leads to messy, unmaintainable code.
- ▸ActionScript executes when the playhead hits the keyframe it's on. A small, lower-case ’a’ symbol appears in keyframes with ActionScript on them.

Walkthrough 2: Creating User Defined Functions

In this walkthrough, you will perform the following tasks:

- ►Create a function that returns a vlaue
- ►Create a function that does not return a value

Summary

- ▶There are three types of non-visual data used by ActionScript: `String`, `Number`, and `Boolean.`
- ▶Only `String` data may be displayed as text in a `TextField`, `Number` data must first be cast to the `String` data type.
- ▶Data retrieved from a `TextField` is typed as a `String`, even if it appears to be a number. It must be cast to the Number type to use it mathematically.
- ▶The `System.capabilities` object support multiple properties, which may be examined to determine what the current device can or cannot do.
- ▶Arguments (parameters) passed to functions exist as local variable inside the function as it runs.
- ▶Functions may or may not return a value. If not, the function return type is `Void`. If so, the function return type is the data type of the value returned.
- ▶ActionScript may be placed on a keyframe, in an external script file, in an external `class` file, or on a `Button` or `MovieClip` instance.