

Action Script Lecture 5 – Using Functions in Flash (Lesson 5) R. Berdan

Objectives:

- 1) Complete the Dynamic slide show from previous class, explain how to add timer and code so a single button starts and stops the slide show advancing the slides every 5 seconds – see previous lesson handout.
- 2) Introduce Functions, naming function, 2 types of syntax used to create a function
- 3) Calling a function
- 4) Parameters, passing parameters to a function (local vs global variables scope)
- 5) Return statement – used to return a value to where it was called
- 6) Functions can call other functions even themselves – recursive function

A function is a group of statements enclosed within braces and given a name. Essentially functions are miniature programs you can call when pressing a button or when the movie loads. Function names follow the same rules as variables: no spaces, no symbols, no reserved words, no numbers that precede – usually you try to give them unique names that describe what the function does e.g. ScrollMessage().

- 1) Syntax – always make sure function uses lower case f not capital F!

```
function functionName(parameter1, parameter2, ...)  
{  
  
statements  
  
}
```

parameters, are variables or data that can be passed to the function and processed. Some functions have no parameters, others can have several.

To define a function using Flash Actions script in normal mode you would

Actions>User Defined Functions>function to add syntax

- 2) Another way to create a function that can only be input using Expert mode in Flash

```
MyFunction = function (parameter1, parameter 2)  
{  
statements;  
}
```

E.g. last week

```
_root.onEnterFrame = function ()  
{  
statements  
}
```

The latter method is often used to attach custom methods (functions) to objects in Flash

Calling a function with no parameters you simply type the function name e.g.

```
FunctionName()
```

This statement tells Flash to execute all the statements inside the function

If a function contains parameters you simply call it:

```
FunctionName(parameter1, parameter2, parameter3)
```

In both cases above it is assumed the function is in the same timeline – if not you will need to add a target path to the function as follows:

```
_root.clip1.clip2.myFunction()
```

Functions can also be called dynamically based on a value:

```
Eg. _root.VariableName();
```

Exercise 1) Open television.fla in the Lesson 05 /Assets folder

Add the following script to the first frame in the actions layer

```
tvPower = false; // create a variable name tvPower and set it to false i.e. power is off
```

```
function togglePower() // create a function call it toggle power, it has no parameters
{
    if (tvPower) // check if tvPower == true, then turn it off with the statements below
    {
        newChannel = 0; // set channel to 0 – which does not exist i.e. nothing
        tvPower=false; // turn power off if it was on
    }

    else // if tvPower was false then make it true, i.e. turn the power on
    {
        newChannel =1;
        tvPower=true;
    }

    tv.screen.gotoAndStop(newChannel + 1) // target tv movie clip, screen movie clip within
    remote.light.play() // turns red light movie clip on
}
```

by calling the function with a button you can make a single button turn the movie on and if it is already on – to turn the movie off. This negates the use of separate on and off buttons.

To call the function attach the following script to the power button on the remote control

```
on (release)
{
    _root.togglePower(); // need to add the target to the main timeline since that is where the
                        // function is in the first frame
}
```

Adding Parameters to a Function

For example

```
function converttoMoonWeight(myWeight)
{
    weightonMoon = myWeight/6.04;
}
```

call the function with this line

```
converttoMoonWeight(165) // 165 is the parameter that will be passed to the function
```

this could have been a variable entered into a form box - Lets try this make a small program where the user enters their weight, clicks on a button and the moon weight is output to another text box.

Answer

1. Create a new movie, add two text boxes, top one input text, the lower one is dynamic text. Beside the top box place Enter your weight, put text beside the lower box "Weight on the Moon".
2. Give the top box the instance name "myweight"
3. Give the bottom text box the instance name moonweight
4. Drag a button onto the screen
5. Select the first frame of the movie clip and add the following code:

```
function converttoMoonweight(myweight)
{
    weightonMoon = myweight/6.04; // expression set it equal to weightonMoon variable
    moonweight.text = weightonMoon // send the result to the lower text box
}
```

6. Select the button you added to the screen, right click select actions and add the following code:

```
on (press)
{
    myweight = myweight.text // take value entered in text box and set it equal to myweight variable
    converttoMoonweight(myweight) // call function and pass parameter myweight to the function
}
```

Note if you enter 165 into the movie it will return a value of 27.3178807947 to round number to two decimal places you could add the following code.

```
weightonMoon=Math.round(myWeight/6.04); // output for 165 a value of 27
```

When a function is called a temporary array is created called arguments that contains all the parameters passed to it. To view the number of arguments and the value passed to the above function add the following code.

```
function converttoMoonweight(myWeight)
{
    weightonMoon=Math.round((myWeight/6.04));
    moonweight.text = weightonMoon;
    trace(arguments.length) // outputs a value of 1 argument passed to function
    trace(arguments[0]) // outputs to the window 165
}
```

Another example would be:

```
function traceNames()
{
    trace(arguments.length)
    trace(arguments[0]);
    trace(arguments[1]);
}
```

```
traceNames("Robert", "Berdan")
```

output should be arguments = 2, Robert, Berdan

Exercise #2 in Text Open television2.fla in Lesson5/Assets folder

In this exercise you will add a function with a parameter you pass to the function – newChannel. This function will change the tv channel based on the parameter – newChannel

Select Frame 1 of the Actions layer and add the following script to the end of the previous script.

```
function changeTheChannel(newChannel)
{
    if (tvPower)
    {
        currentChannel = newChannel;
        tv.screen.gotoAndStop(newChannel + 1);
        remote.light.play();
    }
}
```

double click the remote movie clip to edit it in place add the following script to the round button 1

```
on (release)
{
    _root.changeTheChannel(1);
}
```

since the function exists on the main time line you must add the target path.

Step 6 will add a call to the function changeTheChannel(parameter) - this is an example of calling a function within another function.

You will then add two more functions channelUp() and channelDown() to the up and down buttons.

```
function channelUp()
{
    if (currentChannel + 1 <= numberOfChannels)
    {
        changeTheChannel(currentChannel + 1)
    }
}
```

Exercise #3 - Using local variable and creating functions that return results (page 165)

If a variable is declared outside a function it is available for use by any function and thus is called a global variable. Variables declared inside a function only exist when the function is called then they disappear – these are called local variables. The advantage of local variables is that when they disappear they free up the memory. Also since they only exist while running within a function you will not run into naming collisions – calling the same variable name by mistake. The visibility of a variable is referred to as its SCOPE in the program.

In this exercise you will create an Array – a variable with multiple values that you will display in a text box when a certain channel is selected.

```
channelNames = [“”, “News”, “Classics”, “Family”, “Cartoon”, “Horror”, “Westerns”]
```

The first value is “” or blank since this channel 0 which does not exist.

```
channelNames[0] = “” or nothing
channelNames[1] = “News”
channelNames[2] = “Classics”
```

You will create the array and add it to the first frame after numberOfChannels=6;

To display the text you will output the value using the following line of code

```
cableBox.cableDisplay.text = displayCableText() // this grabs the return value
```

cableBox is the movie clip name, cableDisplay is the text box name

```
function displayCableText()
{
    if (currentChanel !=0)
    {
        var displayText = "You are viewing " + channelNames[currentChannel] + 1 + "."
    }
    else
    {
        var displayText = ""; // text box is set to blank
    }
    return displayText; // returns value of the channelNames to the text box
}
```

In summary

- 1) you have created and called functions
- 2) you have passed parameters to a function
- 3) you used local variables
- 4) called functions from within other functions
- 5) returned and used results of calling a function