

Functions

Although this tutorial will focus on the power of Functions in ActionScript, it will also touch on two other features that are new in this version of Flash: the Color Object and Loading Variables from an external text file.

Objective

We are going to be creating a dynamic chart in Flash. Its dynamic because the data for the chart exists outside of Flash in a text file and can be changed at any time which will alter the results. This is a Columnar chart and each column will represent a value. In order to show the difference between the data sets, we will use the Color Object to change the color of each Column.

The Data

1. Open up the Text File values.txt
2. You can create a similar text file in any text editor (Note Pad, Word Pad, Simpletext, etc.) Be sure you save the file with a .txt extension and your text editor saves the file in the Text Document format.
3. Here we set up 5 variables for each state. To do this for compatibility with Flash you first declare the variable name, then with the equals sign (=) you set the variable value. If you have multiple variables and values, you separate each with the ampersand character (&). Important. If your value is a long text string that contains characters, make sure that the text does not include an equals sign or ampersand as Flash will interpret those as values and separators respectively.
4. Type in the following into the values.txt file:

```
ma=25&ct=48&vt=99&nh=37&ri=58
```

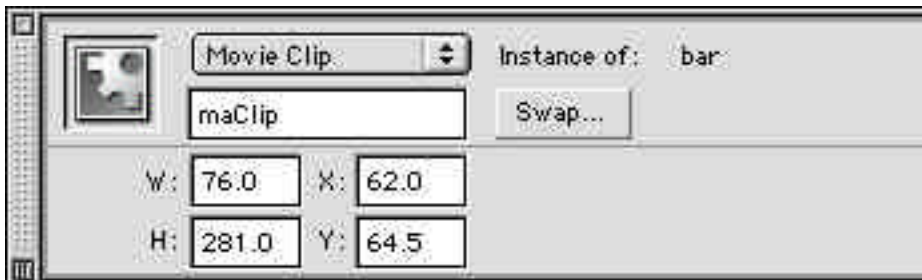
5. Note that there are no spaces here. The values typed in here are predicated on the idea that 100 is the highest value that we are tracking. This will be important for our Flash file. More on that later.
6. Save the file.

Setting Up the Movie

1. Open the Movie Chart.fla
2. The background of the movie simply contains an X/Y axis for the data we are tracking. The X axis is for the various New England states and the Y axis is for the amount of tax dollars each state spent in a given fiscal year.
3. Open the Library. Window > Library.
4. In the Library panel, double click on the bar Movie Clip.
5. The bar is a rectangle with no stroke, but note the position. It is not centered, the bar's bottom left edge is placed on the crosshair. This will be important later when we scale the bar to reflect the data points.



6. Return to Scene 1 by clicking on the Scene 1 tab.
7. Drag an instance of the bar Movie Clip on the stage above the MA label.
8. Use the Info Panel to position the Movie Clip at X: 62 and Y: 342.5.
9. In the Properties Panel give this Instance an Instance Name: maClip.



10. Drag another Instance of the bar Movie Clip onto the stage and position as follows: X: 149.3 and Y: 342.5.
11. Give this an Instance name of ctClip.
12. Repeat the above steps to create 3 more Instances of the bar clip and give them the following X/Y locations and Instance Names:

X/Y Location	13. Instance Name
X: 236.6 and Y: 342.5	14. vtClip
X: 323.8 and Y: 342.5	15. nhClip
X: 411.1 and Y: 342.5	16. riClip

17. Drag an Instance of the btn Button from the Library onto the stage underneath the chart and position it in this location: X: 265 and Y: 423.

Loading Variables

1. Click on Frame 1 of the Actions layer.
2. In the Actions Panel make sure that you are in the Expert Mode by clicking on the icon in the upper right-hand corner and choosing Expert Mode. The Expert Mode will simplify the process of adding scripts to our movie by allowing us to just type them in.
3. Add the following script to load the Variables from the text file you created earlier:

```
loadVariables("values.txt",_level0);
```

4. Values.txt is the text file that we created earlier and _level0 is the root level of the movie. If you were to test the movie at this point nothing would happen and you wouldn't see the Variables in action. What we need to do is bind the Variables with the Movie Clip Instances for each State and then change the scale of each Instance to reflect the data.

Creating the Function

Flash functions are simply a chunk of code that can be reused throughout a program. The basic syntax is as follows:

```
function funcName () {  
    Statements  
}
```

First we declare a function, then give it a name, which could be anything we want. The open and closed parentheses could contain parameters. Parameters can add additional functionality to Functions. Pardon the redundancy there, but that's what they do. Then we open up a set of curly braces and where Statements is indicated, we can run any number of scripts that we want to. The function is no good on its own however, and needs to be called on by a frame or button action. From a button, for example, we could call on the function like so:

```
On (release) {  
    FuncName();  
}
```

Let's create our function which will change the height of the bars using the _yscale property in Flash.

5. Continuing with the Frame 1 script that we already started, type in the next line the following:

```
function changeBars() {  
}
```

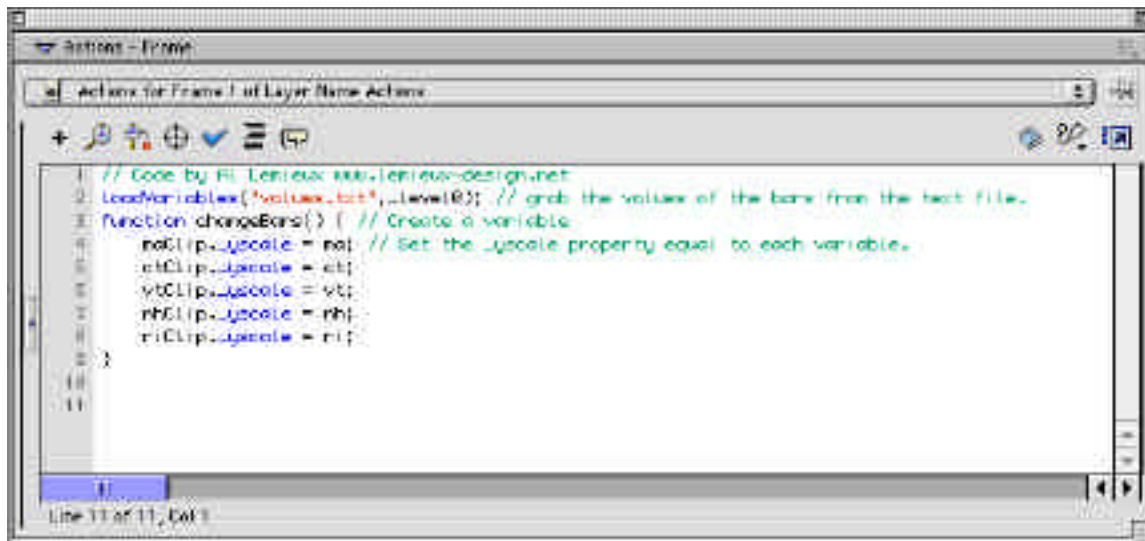
6. Now that we have a function we can start putting statements into the curly braces. Inside the curly braces type in the following lines of code that will change each of the bars height:

```
maClip._yscale = ma;  
ctClip._yscale = ct;  
vtClip._yscale = vt;  
nhClip._yscale = nh;  
riClip._yscale = ri;
```

7. Its not rocket science what we're doing here. Each Movie Clip Instance that we named Is having their height changed to the value of the Variables we pulled in from the text

file. All we need to do that is use the `_yscale` property. Your full script for Frame 1 should be the following:

```
loadVariables("values.txt",_level0);
function changeBars() {
    maClip._yscale = ma;
    ctClip._yscale = ct;
    vtClip._yscale = vt;
    nhClip._yscale = nh;
    riClip._yscale = ri;
}
```



8. If you were to test the Movie at this point, nothing would happen because we haven't called on the function yet.
9. Click on the Get Data button on the stage.
10. In the Actions Panel, type in the following:

```
on (release) {  
    changeBars();  
}
```

11. Now Test the Movie, Control > Test Movie. When you click on the Get Data button, the bars should change in height.

Changing Colors

Now that we can change the height of each Movie Clip Instance with Variables and a Function, its time to work on the color issue. We want to represent each state with a different color. This could be done with multiple versions of the bar Movie Clip, each version being a different color, which would give you 5 Movie Clips to work with in the Library. Why work with 5 Movie Clips when all you need is 1?

The Color Object once defined, can be used to change the color of each Movie Clip Instance. The Color Object is actually a Class and can be used to programmatically dictate the color and transparency of a movie Clip or main movie. Once the Class is created we can change the color by finding out the current color with the getRGB command and then changing the color with the setRGB command.

12. Click on the first Movie Clip Instance above the MA label on the stage.

13. In the Actions Panel create the Color Class as follows:

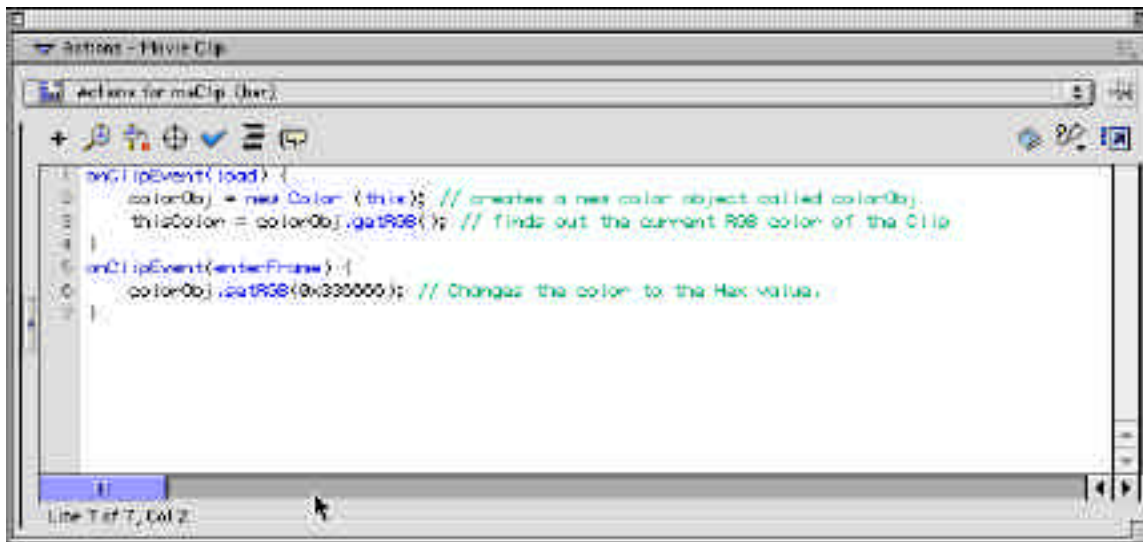
```
onClipEvent(load) {  
    colorObj = new Color (this);  
    thisColor = colorObj.getRGB();  
}
```

14. First of all, I want to address the fact that we are working with a Movie Clip object here so that's where the code is going. Another thing here is the fact that we are creating the Color Class at the loading of the Movie Clip (onClipEvent(load)). This is extremely important so that as soon as the Movie Clip loads, the Class is created and the getRGB command runs. We're not changing the color of the Clip just yet, just getting it. That's what the second line does. It creates a Variable called thisColor and then uses the colorObj.getRGB() command to find out the current RGB values of the clip.

15. Now we can change the color. Add the following statements in the Actions Panel:

```
onClipEvent(enterFrame) {  
    colorObj.setRGB(0x336666);  
}
```

16. OK, so the Movie Clip gets loaded, we find out what its RGB values are and then on the enterFrame event, we change its color. Notice, we are using the colorObj Variable with the setRGB command to change the value. Inside the parentheses, we define the RGB values in Hexadecimal, which if you know HTML, is simple. The first pair of letters or numbers is the Red value, the second pair is the Green value, and the last pair is the Blue value. Even pairs are always web safe colors. Flash's syntax diverts a little here from HTML with the 0x first followed by the Hex color.



17. Change the other Movie Clip Instances with the following code:

ctClip

```
onClipEvent(load) {
    colorObj = new Color (this);
    thisColor = colorObj.getRGB();
}
onClipEvent(enterFrame) {
    colorObj.setRGB(0x99A921);
}
```

vtClip

```
onClipEvent(load) {
    colorObj = new Color (this);
    thisColor = colorObj.getRGB();
}
onClipEvent(enterFrame) {
    colorObj.setRGB(0xCC6600);
}
```

nhClip

```
onClipEvent(load) {
    colorObj = new Color (this);
    thisColor = colorObj.getRGB();
}
onClipEvent(enterFrame) {
    colorObj.setRGB(0xCCFF00);
}
```

riClip

```
onClipEvent(load) {
    colorObj = new Color (this);
    thisColor = colorObj.getRGB();
}
onClipEvent(enterFrame) {
    colorObj.setRGB(0x9966CC);
}
```

```
}
```

18. The only thing that's changing here for each is the Hexadecimal value on the setRGB command.
19. Now when you test the movie, Control > Test Movie, each bar is a different color. If you don't like the colors, you can go back in the code and change them.

Finishing Up

So we've changed the color of our bars with the Color Object, we've changed the height of each Column with a function, and we've imported Variables from an external text file. How is this dynamic?

1. Go to the values.txt file and change the values for each state.
2. Go back to Flash and Test the movie again. The height of each column should change.

Resources: ActionScript the Definitive Guide by Colin Moock. O'Reilly Books. ISBN: 1-56592-852-0.