Flash - Essesntial Programming Concepts

Here is a quick overview to some of the programming concepts you will see commonly used, with this information you should be able to see what's going on if you look at someone else's code.

Variables
These are containers for data. This data can be anything from a person's name to a number that controls the speed of an objects movement.

- You give the variable a name that you can use to reference it in ActionScript
- You give the variable a value that's stored in that named container
- You use the name to extract the value from the container whenever you need it

Example:

```
myName = "Fred";
Print (myName);
```

The single equals sign assigns a value on the right of the equals sign to the variable name that appears on the left of the equals sign.

Use the set variable action under actions in normal mode to set a variable.

Data Types
Number – This is the simplest data type, it defines the variable as a numeric value.

```
x = 90;
y = 8;
```

Numbers can have mathematical operations applied to them x + y and the result is 98.

String – This is a group of characters enclosed by quotation marks.

```
z = "Smelly pants";
x = "90";
y = "8";
```

It can contain either numbers or letters. If you did the equation x + y the result would "908" as the two are spliced together.

Booleans – This is either true or false.

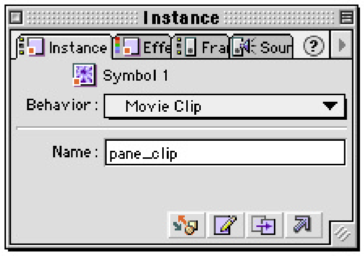
```
isMyNameFred = true;
isMyNameDave = false;
```

They are commonly used to test if a condition exists for example:

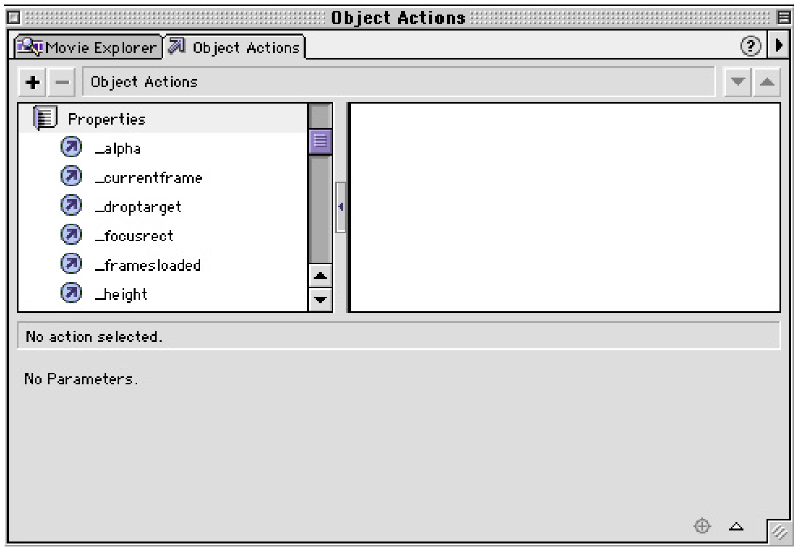
```
if (isPasswordCorrect == true)
then proceed
```

Note the double equals sign this is checking to see if a value is true. A double equals sign is called the equality operator. A single equals sign is setting the value and that's the difference.

Objects and Movie Clips
Objects are groups of properties that have names and values. Things that you build in Flash, especially movie clips are objects. For example you have a movie clip that when it is placed on the stage it is called pane_clip this would inherit all it's properties by default as in _width, _height and _alpha and you can see these by opening the Properties book in the Object Actions window:



Symbol 1
Behavior : Movie Clip
Name : pane_clip



No action selected.
No Parameters.

You can reference these properties and assign their values to variables. You can then assign these variables values to an objects property.

```
alphaValue = 50;
pane_clip._alpha = alphaValue;
```

You can see that this allows information to be passed around in a movie and changed by ActionScript.

Operators

These are symbols that perform some kind of processing on variable.

+	plus
*	times
-	minus
/	divide
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

They work in order of precedence when doing maths calculations so:

1 + 2 * 3 = 7 because 2 * 3 is worked out first then 1 is added to it

(1 + 2) * 3 = 9 because 1 + 2 is worked out, then multiplied by 3

ActionScript Programming Structures

We've looked at the basic concepts now it's time to look at how it's organised into levels of detail:

Expressions
This is a collection of code elements (a collection of variables and operators) that will produce a value:

```
12 * 5
23 + 18
67 – 32
```

Statements
This are generally one line of self contained code that will perform an execution and they are terminated by a semi colon:

```
gotoAndPlay (2);

widthValue = pane_clip._width;
```

Both of these statements are self contained and will achieve a specific outcome. It's vital that when your writing by hand in Expert mode that you insert those semicolons at the end!

Blocks
This is a group of statements contained within braces {}. This contains a set of related statements, all of which need to be run to achieve a desired result:

```
on (press) {
        alphaVal = 50;
        pane_clip._alpha = alphaVal;
}
```

Programming Flow Control

This is the manipulation of what order statements are run and allows us to move away from the straightforward linear structure of running movie clips. We are going to look at two areas branching and looping constructs.

Branching
This is used to make decisions and is used by the if…else structure.

```
if (hitTest(_parent.macload) && !snapBack) {
        snapBack = false;
        play ();
}
else {
        snapBack = true;
}
```

The if and else part of this code have initial conditional clauses so there are two routes to go down. It will never carry out both blocks of code it will always be one or the other.

Looping
Looping allows us to loop code until a condition is met. Key looping structures are:

for
while
do…while

While loop

```
while (I am hungry) {
        sit at the dining table;
        eat some food;
        take a drink;
}
```

Obviously this isn't real code it just shows you an example of what you could set up. It does a certain job until there is a change in that condition's status. Trigger events may include a user's click or a variable reaching a certain number.

For Loop
The conditions to change this loop are contained within it so it doesn't require an external source like the while loop:

```
for (i=0; i<5; i++) {
        print “hello”;
}
```

The first line initialises what is going to happen. You always need 3 parts in the argument:

1. The initial variable of the loop (in this case i) is the loop counter.
2. The condition that will end the loop (here we're telling flash to stop looping when i<5 is no longer true).
3. What to do after running the statements in the block for the loop (here we increment the value by 1 using ++ is short for writing i=i+1).

After that initialisation comes the code block saying we want to print at each iteration of the loop. So it prints hello 5 times.

Do…While Loop
This is similar to the while loop except it always processes it's code block at least once before checking the condition:

```
x=15;
do {
        print “hello”;
}
while (x<10);
```

This would print hello once before ending the loop.

Functions

Whenever you need to identify a chunk of code in multiple locations in a movie you should think “This needs to be a function”. If the code is run frequently it would be stupid to keep rewriting it in multiple locations.

A function is a lump of code with a predefined purpose and which can be called from elsewhere in the movie, the function will do it's job and then return the result to the movie clip or piece of code that called it.

When you create a function, you need to include the names of the variables that will be passed on in the defination statement:

```
function Multiply (a, b) {
        return a*b;
}
```

The word function tells flash that this code will be called whenever someone puts the word Multiply into their code. Then follows the two variables a and b. The result is then returned back in the next line of code.

So to call this function into operation in ActionScript you would use

```
multiply (3, 4);
```

Obviously the numbers can be different at different locations in your code.

A useful rule is that functions should only ever perform one task.