Actionscript 3.0 Syntax Cheat Sheet

Blue: AS3 Reserved Words

Red: Words to replace (defines what should go there)

Black: Words with no AS3 dependency or caracters necessary for code to work

Declaring a variable

```
var VariableName:DataType;  //declared but not initialized
var MyVar:Boolean;          //example of declaring a Boolean
```

Setting a variable's value

```
var VariableName:DataType = value;          VariableName = value;
var MyVar:Boolean = true; //declare & set   MyVar = true;    //already declared
```

Arrays: Creating

```
var myArr:Array = new Array('item1', 2, myVar3);
var myArr:Array = ['item1', 2, myVar3]; //same as previous
```

Manipulating

```
myArr.push(4); //puts the value 4 on the end of the array
myArr.shift(); //removes the first item from the array
```

Accessing

```
ArrayName[index]; //replaced by the item in the array at that index
trace( myArr[0] ); //the 0 index is always the first item in the array
trace( myArr[ myArr.length-1 ] ); //access the last item in the array
```

Objects: Creating and Manipulating

```
var myObj:Object = new Object();
myObj.myParam = 'my value'; //create as many properties as you want
var myObj:Object = {myParam:'myValue'}; //same as previous
```

Accessing

```
myObj.parameter; //if the parameter does not exist it returns undefined
trace( myObj.myParam ); //access the value of myParam: 'my value'
```

Loops

```
while(condition) { //Loop continues while the condition is true and ends when it is false }
var i:int = 0;
while(i < 10) {
        i++ //Once i increments to 10 the loop condition will be false
}

for(initializer; condition; counter) { //Loop continues until the condition is false
for(var i:int = 0; i < 10; i++) { //value of i each iteration: 0,1,2,3,4,5,6,7,8,9 }
```

The Display List: Adding Items

```
MovieClipName.addChild(MovieClipNameToAdd); //Remember that the Display List is just an Array
this.addChild(MyMovieClip); //assuming this is within a MovieClip class, or on the Timeline
this.addChildAt(MyMovieClip, 0); //Objects at a lower index are behind items at a higher index
```

Moving Items

```
this.setChildIndex(MyMovieClip, 2); //Moves the clip to index 2
this.swapChildren(MyChildClip1, MyChildClip2); //Swaps the two child clips.
this.swapChildrenAt(0, 1); //Swaps the objects at index 0 and 1
```

Removing Items

```
ParentClipName.removeChild(MovieClipNameToRemove);
this.removeChild(MyMovieClip);
```

Operators

```
+  //add
-  //subtract
*  //multiply
/  //divide
%  //modulo

=  //assignment

== //equal to
!= //not equal to
<  //less than
<= //less or equal
>  //greater than
>= //greater or equal

!  //not
&& //and
|| //or
```

Conditionals (Evaluations of True or False)

```
if(true) { //will run }

if(false){ //will not run }

if(isDone) { //if isDone is true
        //code here will run
} else if(!isLoaded) { //if isLoaded is not true
        //code here will run
} else { //otherwise code here will run }

if(myString == 'Hello') {}  //string comparison
if(myNum < 10) {}           //number comparison
if(!isDone) {}              //logical NOT
if(myNum == 0 && isDone) {} //logical AND
if(myNum == 0 || isDone) {} //logical OR
```

Math

increment and decrement

```
i++; //same as i += 1; //same as i = i+1;
i--; //same as i -= 1; //same as i = i-1;
```

calculations and operations

```
i = 10*10; //100 i = 10/10; //1 i = 10%10; //0
i = Math.random();   //number between 0 and 1
i = Math.floor(1.8); //1 always rounds down
i = Math.round(1.5); //2 normal math rounding
i = Math.ceil(1.1);  //2 always rounds up
```

Actionscript 3.0 Syntax Cheat Sheet

Creating Functions: simplest (no parameters)

```
function functionName():returnType {
	//function content
}

function myFunction():void{                    //remember that a function only runs when called
	trace('You just called myFunction');
}
```

with parameters and return

```
function functionName(param1:DataType, param2:Datatype, ...):returnType {
	return ValueOfReturnType;  //return something unless the returnType is void
}

function addNums(num1:Number, num2:Number):Number{   //the returnType is Number here, so we return
	var sum:Number = num1 + num2;                    sum which is of type Number.
	return sum;
}
```

Calling Functions

```
functionName(param1:DataType, param2:Datatype, ...);
myFunction();                    //causes the code defined in myFunction to run.
var sum:Number = addNums(2, 4); //addNums runs and is replaced with whatever value it returned.
```

Listening for an event

```
DisplayObjectToListenTo.addEventListener(EventType.EVENT, onEventHandled);
stage.addEventListener(MouseEvent.CLICK, onMouseClicked);
```

Handling an event with an event callback function

```
function onEventHandled(e:EventType):returnType { //will run whenever the event being listened
	//function content                             //for happens.
}

function onMouseClicked(e:MouseEvent):void {       //Based on the type of event being listened for
	trace(e.stageX, e.stageY);                     //the event object returned may have special
}                                                  //properties, such as mouse coordinates.
```

OOP: Classes

```
package {
	public class ClassName { //Class Name must match name of .as file EXACTLY

		public var property:DataType; //Public Property

		public function ClassName(param1:DataType, ...) { //Constructor (same name as Class)
			//Code here runs when a new object of this class is instantiated.
		}

		public function MethodName(param1:DataType, ...) {} //Public Method
	}
}

package {
	public class MathEngine {

		public var sum:Number;

		public function MathEngine() {
			sum = 0;
		}

		public function AddNums(num1:Number, num2:Number):void {
			sum = num1 + num2;
		}
	}
}
```

Instantiating Class Objects

```
var varName:ClassName = new ClassName();
var ME:MathEngine = new MathEngine(); //Creates a new MathEngine Object (constructor runs here)
```

Using Class Objects

```
varName.MethodName(param1, ...);  //or  varName.ParameterName;
ME.AddNums(4, 5);  //Calls the AddNums method defined in the MathEngine class
trace(ME.sum);     //Accesses the sum parameter defined in the MathEngine class
```