CS 116 – Action Script Types & Variables

Elie Abi Chahine

Types

Numeric:

ActionScript 3.0 includes three specific data types for numeric data:

- **Number**: any numeric value, including values with or without a fraction
- **int**: an integer (a whole number without a fraction)
- **uint**: an “unsigned” integer, meaning a whole number that can’t be negative

Boolean:
A true-or-false value, such as whether a switch is on or whether two values are equal

Types

- The simple data types represent a single piece of information: for example, a single number or a single sequence of text.
- However, the majority of the data types defined in ActionScript could be described as complex data types (aka Classes), because they represent a set of values grouped together.
- For example, a variable with the data type Date represents a single value—a moment in time. Nevertheless, that date value is actually represented as several values: the day, month, year, hours, minutes, seconds, and so on, all of which are individual numbers. So while we think of a date as a single value (and we can treat it as a single value by creating a Date variable), internally the computer thinks of it as a group of several values that, put together, define a single date.

Major Types

The primitive data types include: Boolean, int, uint, Number, String, Null, and void.

The ActionScript core classes also define the following complex data types: Object, Array, Date, Error, Function, RegExp, XML, and XMLList.

Note: To get a list of all the types you can create do the following:
- In the Actions window click on the "+" sign (Add new item to the list)
- Go to "Top Level"

Boolean

The Boolean data type comprises two values: **true** and **false**.

```
Eg:    var bName:Boolean = true;

       var bName:Boolean = Boolean(true);
```

Note: **No other values are valid for variables of Boolean type.**
The default value of a Boolean variable that has been declared but not initialized is false.

int

The int data type is stored internally as a 32-bit integer and comprises the set of integers from -2,147,483,648 to 2,147,483,647.

If your variable will not use floating-point numbers, using the int data type instead of the Number data type should be faster and more efficient.

```
Eg:    var iName:int = 10;

       var iName:int = int(10.5);   /* iName will be equal
                                       to 10 */
```

Note: **The default value for variables that are of the data type int is 0.**
For integer values outside the range of the minimum and maximum int values, use the Number data type, which can handle values between positive and negative 9,007,199,254,740,992 (53-bit integer values).

uint

The uint data type is stored internally as a 32-bit unsigned integer and comprises the set of integers from 0 to 4,294,967,295 (so only non-negative numbers).

For example, you must use the uint data type to represent pixel color values.

```
Eg: var uiName:uint = 10;

    var uiName:uint = uint(10.5);  /* uiName will be equal to 10 */

    var uiName:uint = -10;  /* Will give you a warning but will
                              assign the value 4294967286 to the
                              variable */
```

Note: **The default value for variables that are of the data type uint is 0**

Number

The Number data type can represent integers, unsigned integers, and floating-point numbers.

However, to maximize performance, you should use the Number data type only for integer values larger than the 32-bit int and uint types can store or for floating-point numbers.

```
Eg: var nName:Number = 10;

    var nName:Number = Number(10.5);  /* nName will be equal to 10.5 */

    var nName:Number = -10.5;
```

Note: The default value for variables that are of the data type Number is "NaN". Also "NaN" is the result of any operation that should return a number but does not. For example, if you attempt to calculate the square root of a negative number, the result will be NaN. Other special Number values include positive infinity and negative infinity.

String

The String data type represents a sequence of characters.

```
Eg:
    var sName:String = “Hello”;

    /* sName will be equal to Hello */
    var sName:String = String(“Hello”);

    /* “ World” will be added to the string which will lead to
        sName = “Hello World” */
    sName += “ World”;
```

Note: **The default value for a variable declared with the String data type is null. The value null is not the same as the empty string (""), even though they both represent the absence of any characters.**

Array

An array is a programming element that acts as a container for a set of items.

```
Eg: var aName:Array = Array(10); /* This will contain an empty
                                    array  with 10 slots */

    var aName:Array = [0,1,2,3,4,5,6,7,8,9];

    var aName:Array = [1,2,3,4,5,6,7,"blah",8];
```

Indexing an array is done by using the [ ] operator.

```
Eg: var aName:Array = [12,21,32,45,5,62,79,18,39];
    aName[3] = 17; /* Now the array looks like this
                      [12,21,32,17,5,62,79,18,39]; */
```

Note: More on the array will be covered in a separate chapter.

Null

The Null data type contains only one value, null. This is the default value for the String data type and all classes that define complex data types, including the Object class.

None of the other primitive data types, such as Boolean, Number, int and uint, contain the value null. Flash Player will convert the value null to the appropriate default value if you attempt to assign null to variables of type Boolean, Number, int, or uint. You cannot use this data type as a type annotation.

Eg:
```
var iName:int = null;       /* Name will be equal to 0*/
var sName:String = null;    /* Name will be equal to null */
var bName:Boolean = null;   /* Name will be equal to false */
```

*

The '*' data type is there for users to create a untyped variables, which is equivalent to omitting a type annotation.

Eg:
```
var Name:*;             /* same as var Name; */
var Name = 5;           /* Name will be equal to 5 */
var Name = "Blah";      /* Now Name contains a string "Blah"*/
```

Note: **Only untyped variables can hold the value "undefined".**

Default Values

Data type	Default value .
Boolean	false
int	0
Number	NaN
String	null
uint	0
Not declared (equivalent to type annotation *)	undefined
All other classes, including user-defined classes.	null

Variables

How to create variables:

var VariableName:Type

- First we need to use the Keyword: **var**
- Second we put the name of the variable
- Third we specify the type

Eg: var nBlah:Number;

Note: Not using the *var* keyword will result with a compiler error

Naming Rules

- You have to specify a unique name each time you create a variable (taking into consideration that ActionScript is case sensitive).
- Variable names can contain the following characters [a-z] [A-Z] [0-9] "$"and the "_" no other characters are allowed.
- You can start the name with [a-z] [A-Z] "$" or "_" but never with a digit ([0-9])
- The Variable name can't be a keyword found in ActionScript (anything that turns blue is off limit)
- Eg: Blah , _Blah , blah , Blah_1 , B1lah , $Blah , B$lah (Are all good)
  1Blah , #Blah , int , Number (Are all illegal)

Keywords & Reserved words

Keywords:

as	break	case	catch	class
const	continue	default	delete	do
else	extends	false	finally	for
function	if	implements	import	in
Instanceof	interface	internal	is	native
new	null	package	private	protected
public	return	super	switch	this
throw	to	true	try	typeof
use	var	void	while	with

Don't worry, you don't have to memorize them. They turn to blue when you use them in the editor.

Keywords & Reserved words

Syntatic Keywords:

each	get	set	namespace
include	dynamic	final	native
override	static		

Reserved words:

abstract	boolean	byte	cast	char debugger
double	enum	export	float	
goto	intrinsic	long	prototype	short
synchronized	throws	to	transient	type
virtual	volatile			

Please, do not memorize those words. Just make sure you name your variable with a little bit of thought and you will be fine.

Variables

Guides to writing good code:

- Naming conventions specify the rules used to make up a name in your program. A variable name labels an object and conveys information about its purpose.
- Naming conventions make it easier for yourselves and me to understand your code. It also makes it easier to understand and reuse the code for an assignment later in the semester.
- Here are some useful guidelines:
  - Create a name that is informative, concise, memorable, and if possible pronounceable.
  - When naming a variable, start with one lowercase letter that will relate to the variables type, then a name that relates with it's functionality and use:

var iEnemies_Num:int;

Variables

<u>Guides to writing good code:</u>

- Another example, an automatic variable keeping track of the number of vertices of a 3d object, might be declared as:

```
var iPoints:int;                /* ok */
var iNum_Points:int;            /* ok */
var iNumberOfPoints:int;        /* probably too much */
```

What lowercase letter to start with when using different types?

Array	‘a’
int	‘i’
uint	‘ui’
Boolean	‘b’

I think you got the point !!!

Variables

Guides to writing good code:

- Variables that can have very short names are the ones used as indices in iterations. The use of i and j for loop indices, and s and t for strings is so frequent that there is little to be gained and perhaps some loss in longer names. Compare:

```
for (arrayIndex = 0; arrayIndex < numberOfElements; arrayIndex++)
{
  elementArray[arrayIndex] = elementArray;
}
```

To

```
for (i = 0; i < nsize; i++)
{
    elem[i] = i;
}
```

The End ☺