**Variables: Data Types**

As mentioned earlier, variables can contain many kinds of data or information. There are several official TYPES of data which JavaScript can understand. In brief, I am going to outline a few of the most important of these data types, since I will be referring to them when we start coding a little later.

In JavaScript, there are three essential "primitive" data types: numbers, strings, and booleans.

Here are some numbers:

10

-147

.0035

1.5

-520.0003

As far as JavaScript is concerned, numbers may be positive or negative, and may have decimal places or be without decimal places.

As I mentioned earlier, however, JavaScript is a C-based language; the C-based languages include C++, Java, and JavaScript, among others. In C-based languages, numbers are divided into two separate varieties: integers (int) and floats (float).

Integers (int) are whole numbers with NO decimal places. Here are some integers:

1

-100

247

-68000

0

Floats (float) are numbers WITH decimal places. Here are some floats:

.0003

146.5

-2.575

10000.1

Again, when you are programming in JavaScript, you will not usually differentiate between integers and floats. However, JavaScript DOES recognize a difference between integers and floats, and it has a few built-in features which allow you to manipulate numbers as either one or the other.

STRINGS are collections of alphanumeric characters. Anything you can type on a computer keyboard may potentially be part of a string. Strings, in JavaScript, are always marked with quote marks. Here are some strings:

"Hi"

"Wow, you have 2.5 children!"

"Michael"

"Hey, dude, I like your green hair..."

As you can see, strings are ALWAYS enclosed in double-quote marks, and may contain a variety of characters, whether text, numbers, spaces, or punctuation.

In HTML, you have already dealt with strings frequently. Every time you set an attribute for a tag, you are actually setting that attribute equal to some string value (i.e. **src="capitalA.gif"** or **width="54"**).

Even though strings may have numbers contained within them, there is a distinct difference between string-based numbers and actual numbers. For instance, these two values are NOT the same:

1

"1"

The first example is the actual number, 1. The second example is the CHARACTER "1", which is NOT a number; rather, it is merely a symbol that you type on the computer keyboard. You can NOT perform mathematical computations on a STRING "1", whereas you CAN perform mathematical computations on a NUMBER 1.

Having said this, JavaScript has on-the-fly conversion capabilities which automatically convert string numerical characters to numbers, as well as numbers to strings, under appropriate conditions. Although this sort of automatic conversion is a tremendous convenience, it does NOT keep variables from being EITHER numbers OR strings at any given time; you will frequently be called upon to differentiate between these two data types in your code. I myself have been tripped up more than once by these on-the-fly conversions while programming in JavaScript, so I just wanted to make sure that I mentioned this feature.

The third "primitive" data type in JavaScript is BOOLEAN. Booleans sound very technical, but they are quite simple. Invented by George Bool in a previous century, boolean numbers, in JavaScript and other traditional programming languages, have just two states: false and true, 0 and 1, off and on. The values of false, 0, and off are all equivalent, while the values of true, 1, and on are also equivalent. In JavaScript, booleans will always be either false or true (not off/on or 0/1).

Booleans are everywhere in computer-based or electronics-based equipment. Have you ever noticed the | and 0 characters printed on the on/off switches for your computer, stereo, or other pieces of electronics equipment? Those characters represent on and off for the switch, with | or 1 being ON, and 0 being OFF. That's a BOOLEAN! Switches themselves are firmly embedded in the history of the first computers.

Once upon a time, computers were just gigantic, room-sized banks of switches, and, later, switches and lightbulbs, which turned on and off mechanically; when a lightbulb or switch was on, that was a 1, whereas when a lightbulb or switch was off, that counted as a 0. Using these 0's and 1's, you could create BINARY code (binary means numbers in base 2, which is all 0's and 1's). Just like the secret codes you may have played with as a child, you could say that the number 0 represents the letter "a", and that the number 1 represents the letter "b", and that the number 10 (10 in binary is 2 in our regular counting system) represents the letter "c", and that the number 11 represents the letter "d", etc. With big enough binary numbers, you could extend your binary "secret code" to represent not only letters, but also colors, or sound waves, or picture information, or whatever you like. Once you have decided which binary numbers represent which characters or sounds or pictures, you could store this binary information in your computer. With the "key" to your binary "secret code", you could then take this binary information out of your computer, manipulate it in some fashion, convert it back into a semblance of its original form, and display, play, or printout this media.

Using switches and lightbulbs, of course, computers could only store and manipulate a VERY limited amount of binary information (it took roomfuls of switches just to get enough numbers together to add up your checkbook, let alone represent millions of colors!); those early computers also weren't very fast. Now, with silicon microchips and advanced magnetic memory, computers can store and manipulate mind-boggling quantities of information at magical speeds. All of this data, however, is still encoded, stored, and manipulated in the computer using on/off, binary, BOOLEAN states.

Although you will probably never have to get deep enough into the computer to be forced to look at actual binary conversion of data (thank goodness!), JavaScript, naturally enough, is still heavily reliant on booleans and boolean states. You will frequently be testing conditions in code to see whether something is true or false, a boolean state. Has the user clicked on a particular button? That information would be either true (the user HAS clicked on the button) or false (the user has NOT clicked on the button), a boolean state. Has the user visited this page three times? If they HAVE visited the page three times, the condition would be TRUE, whereas if they have visited the page FEWER than three times or MORE than three times, the condition would be FALSE, another boolean state. In some kind of game code, you might have a boolean variable which remains false until the user has collected all of the "magic wands" in the game (for instance), whereupon the boolean variable would be changed to true, allowing new levels of the game to open up. Boolean variables are everywhere in JavaScript.

Besides the three "primitive" data types, there is also the OBJECT data type in JavaScript. Objects include a whole host of structures and entities which are beyond the scope of this introductory lecture. In the next section, however, we will discuss the basic features of generic objects, as you will need to know a little bit about objects in order to use JavaScript effectively (since almost everything in JavaScript is built using objects).

Beyond the primitive data types, and objects, there are two more data types you should be aware of: null and undefined.

The variable that is null contains NO information. You should know, however, that null is NOT 0; Zero (0) is an integer, a number. The null variable is empty.

In JavaScript, the null variable is differentiated from the undefined variable. An undefined variable has not been declared, and therefore does not exist, or has not been initialized, and so contains no data.

You will find that there are many times when you will need to check for the null or undefined variable data types, especially when handling user feedback or input. For instance, if you program JavaScript to bring up a prompt window (which allows a user to type in some information and click either the OK or Cancel buttons), you will need to test for the undefined or null variable if a user has not given any input or has pressed the "Cancel" button on the prompt dialog box; this is just one place where you might need to test for this data type.

The null variable and the undefined variable may be treated interchangeably in JavaScript, but can be detected individually in the most modern browsers, if needed.

There are many other data types, but they are beyond the scope of this class. For more information on data types, refer to *"JavaScript: The Definitive Guide"*, Chapter 3.