Chapter 3
Data Types and Variables

Adapted from
JavaScript: The Complete Reference 2nd Edition
by
Thomas Powell & Fritz Schneider

Jargon Review

- *Variable* – “container” to hold data in a script
- *Identifier* – the “name” of a variable
- *Literal* – data that appears directly in source code (e.g. 5, “thomas”, true)
- Type – describes the format/nature of the data
  - Primitive JavaScript types: string, number, Boolean
  - Complex JavaScript types: function, array, object

Jargon Review Contd.

- There are strong, weak, and untyped languages
  - JavaScript is a weakly typed language for better or worse
- When you *declare* a variable you are setting up space to hold data
  - `var x;`
- When you *assign* a variable you put some data into it (associate it with something)
  - `x = "foo";`

Variables

- Variables are defined with the **var** keyword
  - `var myName = "Fred";`
  - `var x;`
  - `var x,y,z;`
  - `var a = 1, x="hello", longerName;`
- Variable names or more appropriate identifiers should start with a letter and may be followed by any number of letters, digits, underscores, or dashes
  - Variable identifiers must not contain special characters or white space

Numbers

- JavaScript doesn't make great distinction to the type of numbers unlike some languages
- Numbers are represented in 64bits though with integers you should understand that it may be a 32bit representation
- Ranges for JavaScript are typically:
  - Floating point range: $-2.2250 \times 10^{-308} + 1.7976 \times 10^{308}$
  - Integer Range –2147483648 - 2147483647

Numbers Contd.

- Format of numbers include:
  - DecimalDigits.(DecimalDigits)(Exponent)
  - .DecimalDigits(Exponent)
  - DecimalDigits(Exponent)
- All the following are valid numbers in JavaScript
  - 10, -2.71, .4444e7, -1.7E12, 3.5E-5, 128E+100
- The following are not numbers
  - 2,717, 22e100.5, 128e5e10, e10
- Do not use leading zeros in JavaScript numbers since it has special meaning discussed next

Hex and Octal Literals

- You can specify base-16 (Hex) and base-8 (octal) numbers in JavaScript
- Hex has digits 0-9 and A – F (10-15)
  - Format 0x(Hex digits)
  - Examples: 0x0, 0xF8F00, 0xFFFFFF
  - Value is not obvious on Web except color representations
- Octal is not part of ECMA-262 spec but widely supported by JavaScript implementations
  - Format 0(octal digits 0 –7)
  - Examples 00, 0777, 0245

Special Number Values

- These special values are both useful and occasionally troublesome

Property	Value
Number.MAX_VALUE	Largest magnitude representable
Number.MIN_VALUE	Smallest magnitude representable
Number.POSITIVE_INFINITY	Infinity
Number.NEGATIVE_INFINITY	Negative Infinity
Number.NaN	Not a number

Note: You may end up with these values when you do something extreme (error case)

Strings

- In JavaScript a string is surrounded by either single or double quotes (‘t’, “Thomas”)
  - The choice of quotes is interchangeable in the base case but there is reason to be aware of which to use and when

- Strings like numbers are associated with an underlying **String** object.

```
var myString = new String("Thomas");
alert(myString.charAt(2));
alert(myString.length);
```

- JavaScript string are not thought of as arrays though (ex. alert(myString[1])

Special Characters and Strings

- You can add special characters to strings using the following escape codes

Escape Code	Value
\b	Backspace
\t	Tab
\n	Linefeed
\v	Vertical Tab
\f	Form feed
\r	Carriage return
\”	Double quote
\’	Single quote
\\	Backslash

Note: Because of final display environment from JavaScript is often an (X)HTML based web page, escape codes may not always produce intended results because of whitespace rules.

Booleans

- Booleans are the logical data type and take on a value of **true** or **false**

- Often used in conditionals and loops

```
var doincrement = true;
while (doincrement)  { }
```

- Some languages assume 0 = false and 1 = true being used interchangeably. While conversion may make this seem true, it isn't. Non-zero = true and 0 = false, but try not to assume numeric values when using Booleans.

Undefined and Null

- The undefined type is used for variables or object properties that either do not exist or have not been assigned a value

- The null value indicates an empty or non-existent value
  - Placeholder for “nothing”
  - `typeof null` returns object

- Null and undefined aren’t really the same thing so be careful. However, note that they will compare as equal!

Composite Types

- An object is a composite type that can contain primitive and composite types, including functions and other objects
  - The members of an object are called *properties*
  - The member functions are called *methods*

- Properties are accessed with the "." operator
  - `navigator.appName`

- Methods are accessed the same way except you add ( )s since a function is being called
  - `window.close( );`
  - `document.write("Hi there");`

Objects

- We'll see in JavaScript there are four built-in object types
  1. Type related – String, Number, Boolean, Array, Object
  2. Built-in – Date, Math, RegExp
  3. Browser – Window, Navigator
  4. Document – Document, Form, Image, etc.

Objects

- A fifth type would be user created, but that is somewhat rare given the coding style used by many JS programmers as well as the applicability of OOP principles to extremely small scripts.

- In general objects are created with the **new** operator as well as the appropriate object constructor
  -
```
var myLocation = new Object();
myLocation.city = "San Diego";
myLocation.state = "California";
```

Arrays

- An array is an ordered list that can contain primitive and complex data types.
- Array can be declared explicitly or created using the Array object constructor (Arrays are objects too!)
  - `var myArray = [5,67,45243, "hello"];`
  - `var myArray2 = new Array();  // var myArray2 = [ ];`
- Arrays are zero based indexed
- Arrays elements are accessed with the [ ] operator
  - `alert myArray[0];  alert(myArray[1]);`
    `alert(myArray[3]);`
  - `alert(myArray[4]);`
  - `myArray[4] = "hi there";`

Arrays

- As we can see with assigning to new locations, arrays are extended automatically
  - `myArray[500] = "wow!";`
  - Sparse arrays may result! Be careful about memory issues even though JavaScript does use garbage collection.

- Arrays are objects and they have many properties and methods
  - `alert(myArray.length);`
  - `myArray.reverse();`

- See Chapter 7 for details on the numerous properties and methods for Array.

Functions

- Functions are objects and thus are a data type
- Function are declared like so:

```
function functioname(parameter list)
 {
   function statement(s)
   optional return statement(s)
  }
```

- Functions are called by invoking function name with a list of parameters

```
functioname(parameter1, … parameter n);
```

- We wait until Chapter 5 to study functions in depth

Variable scope

- The scope of a variable is the part of a script where it is visible (known)
- Two scopes
  - global [document wide]
  - local [limited to defining function]
- The var keyword when used in a function creates a local variable.
  - We will revisit this topic in depth later on

Type Conversion

- Converting one type of data to another is both useful and a source of numerous errors in JavaScript
  - `var x = "10" – 2; // result is 8`
  - `var x = "2" - "2"; // result is 0`
  - `var x  = "2" + "2"; // result ="22"`
  -
```
var BooleanData = true;
var numericData = 3.14
alert( numericData >= BooleanData)
```

Type Conversion

- You may run into some serious problems with equality (==) thus you should use the type equality operator (===) and consider performing type checks using the **typeof** operator
- The next few slides show the basic type conversions you should be aware of.

Conversion to Boolean Table

Type	Converted to Boolean
Undefined	False
Null	False
Number	False if 0 or Nan otherwise true
String	False if string is empty (length == 0) otherwise true
Other objects	true

Conversion to Number Table

Type	Converted to Number
Undefined	NaN
Null	0
Boolean	1 if true, 0 if false
String	The number in the string or NaN if no number within string
Other objects	NaN

Conversion to String Table

Type	Converted to String
Undefined	“undefined”
Null	“null”
Boolean	“true” if true, “false” if false
String	“NaN”, “0”, or a string representing the number
Other objects	Value of the object’s toString() method otherwise “undefined”

typeof Operator Values

- You should be aware of the values returned by `typeof` when checking conversions

Type	Result of typeof
Undefined	“undefined”
Null	“object”
Boolean	“boolean”
Number	“number”
String	“string”
Object	“object”
Function	“function”

Summary

- JavaScript provides five primitive data types: number, string, Boolean, undefined, and null
- Composite (or if you like complex) data types include objects, arrays, and functions, but underneath everything tends to be an object
- Two primary scopes exist (global and local)
- The language is weakly typed and conversion should be well understood