CGS 3066: Spring 2015
JavaScript Reference

Can also be used as a study guide. Only covers topics discussed in class.

1 Introduction

JavaScript is a scripting language produced by Netscape for use within HTML pages to introduce an element of dynamism to static HTML pages. JavaScript is loosely based on Java and it is built into all the major modern browsers. However, JavaScript is not the same as Java. This document gives an introduction to Javascript and should be used in tandem with class slides.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.
- Runs on the client side and is event-driven.
- Can be used to make changes to HTML/CSS, perform client side validation and build small apps.
- When used with JavaScript frameworks like jQuery and Angularjs, it can be used to build large scale apps including stand-alone Operating Systems.

1.1 Placement on a page

JavaScript code can be included anywhere in an HTML document. But the following are the most preferred ways to include JavaScript in your HTML file.

- Script in the head section.
- Script in the body section.
- Script in the body and the head sections.
- Script in an external file and then include in the head section.
- JavaScript must be enclosed in the <script>tag.
- There are 2 attributes to the <script>tag: language="javascript" and type="text/javascript".
- JavaScript can be written inline, in script tags or linked from a external file.

1.2 JavaScript Syntax

JavaScript follows the C++/Java syntax.

- JavaScript uses the Unicode character set. Unicode covers (almost) all the characters, punctuations, and symbols in the world.
- JavaScript is a case-sensitive language. This means that language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.
- JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs.
- Semicolons can be ignored if statements are placed on separate lines.
- The JavaScript syntax includes literals (constants), variables, keywords, operators, and comments.
- Comments are C-Style. // for single line and /* —- */

1.3 JavaScript Data Types

JavaScript is a dynamically typed language. This means variables aren't declared with type. They can also be assigned values of different types. However, the different types of data legal in JavaScript are restricted to the following.

- Strings: Any form of textual data.
- Number: Include integers and floats.
- Booleans
- Arrays: Data of the same type, stored in contiguous memory, indexed from 0.
- Objects: Representations of real world entities. Objects have properties(data) and methods (functions) that act on those properties. JavaScript accepts regular objects and JavaScript Object Notation (JSON) objects.
- Undefined: If a variable has been declared but not assigned a value, it's undefined.
- Null: This is used when we want a variable to be defined, but not have any value.

1.4 JavaScript Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword.

```
<script type="text/javascript">
        var money=500;
        var name="Joe";
</script>
```

1.4.1 Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables will have one of two possible scopes.

- **Global Variables:** A global variable has global scope which means it is defined everywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

1.4.2 Variable Names

While naming your variables in JavaScript keep following rules in mind.

- JavaScript reserved keywords can't be used as variable names.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character.
- JavaScript variable names are case sensitive.

1.5 Reserved Words

The following are reserved words in JavaScript. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

1.6 JavaScript Operators

JavaScript offers a variety of operators for arithmetic, comparisons, logical operations, assignment etc.

1.6.1 Arithmetic Operators

These include the basic math operators + (addition), - (subtraction), * (multiplication), / (division) and % (modulus). ++ (increment) and – (decrement) operators also fall under this category. Increment/decrement operators can be done pre-op ++x or post-op x++.

1.6.2 Comparison Operators

JavaScript supports the 6 basic comparison operators. <, <=, >, >=, == and !=. Equality can be one of 2 kinds. == is untyped equality (3.0 and 3 are equal). === is typed equality (3.0 and 3 are not equal). === is also called the identical operator.

1.6.3 Logical operators

Logical operators are typically used in conditions. The result of a logical operation is always a boolean value (true or false). The logical operators provided by JavaScript include AND- &&, OR - || and NOT - !.

1.6.4 Bitwise Operators

These work on individual bits of the data in a variable instead of interpreting the variable as a single data unit. Bitwise operators available in JavaScript are listed below.

- & - Bitwise AND
- | - Bitwise OR
- $\sim$ - Bitwise NOT
- $\wedge$ - Bitwise XOR
- <<- Bitwise Left Shift
- >>- Signed Bitwise Right Shift
- >>>- Unsigned Bitwise Right Shift

1.6.5 Assignment Operators

Used to assign values to a variable. Includes =, +=, -=, *=, /= and %=. x+=y translates to x=x+y.

1.6.6 Miscellaneous Operators

These operators don't fall into any of the above categories.

- Conditional Operator (?): This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.
- typeof: A unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.
- delete: Used to remove objects/ properties of objects from the memory.

2 HTML DOM

The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document. The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects.
- The properties of all HTML elements.
- The methods to access all HTML elements.
- The events for all HTML elements.

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.
The Document object represents the Web page that is loaded in the browser window, and the content displayed on that page, including text and form elements. You can use the methods of the document object to work on a Web page. Here are the most common document methods:

- write() - write a string to the Web page
- open() - opens a new document
- close() - closes the document

JavaScript can use the HTML DOM to find, change, add and delete HTML Elements.

Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML=	Change the inner HTML of an element
element.attribute=	Change the attribute of an HTML element
element.setAttribute(*attribute*,*value*)	Change the attribute of an HTML element
element.style.*property*=	Change the style of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement()	Create an HTML element
document.removeChild()	Remove an HTML element
document.appendChild()	Add an HTML element
document.replaceChild()	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

3 Branches and Loops

3.1 Branches

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true.
- Use else to specify a block of code to be executed, if the same condition is false.

- Use else if to specify a new condition to test, if the first condition is false.
- Use switch to specify many alternative blocks of code to be executed.

3.2 Loops

Loops can execute a block of code as long as a specified condition is true.

- The while loop loops through a block of code as long as a specified condition is true.
- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- The for loop is often the tool you will use when you want to create a loop. It loops through a block of code a number of times.

Examples for loops and branches are on the first 3 JavaScript examples on the course website. More examples can be found on W3Schools and on the Internet in general.

3.3 break and continue

- The break statement, which was briefly introduced with the switch statement, is used to exit a loop early, breaking out of the enclosing curly braces.
- The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block. When a continue statement is encountered, program flow will move to the loop check expression immediately and if condition remain true then it start next iteration otherwise control comes out of the loop.

4 Functions

- JavaScript functions are defined with the function keyword. You can use a function declaration or a function expression.
- Before we use a function we need to define that function. The most common way to define a function in JavaScript is by using the *function* keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.
- Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).
- Functions can be called by other functions, called by code in HTML forms or invoked upon an event trigger, like clicking on a button.

5 Objects

An object is a software representation of a real world entity. An object consists of properties (whose values identify an object) and methods (functions that act on the properties and describe the behavior of the object).

In JavaScript, everything's an object. However, it is not advisable to declare primitive types as objects, since it slows down executions and can have unpredictable side effects.

6 Events

HTML events are "things" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript can react on these events. Often, when events happen, you may want to do something. JavaScript lets you execute code when events are detected. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.
For example,

```
<input type="button" onclick="getElementById('demo').innerHTML=Date()" value="Time">
```

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

- Things that should be done every time a page loads.
- Things that should be done when the page is closed.
- Action that should be performed when a user clicks a button.
- Content that should be verified when a user inputs data.

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly.
- HTML event attributes can call JavaScript functions.
- You can assign your own event handler functions to HTML elements.
- You can prevent events from being sent or being handled.