

A publication of TeachUcomp Incorporated.  
Copyright © **TEACHUCOMP, INC.** 2014

**MASTERING  
INTRODUCTORY  
JAVASCRIPT  
MADE EASY™ V.1.0**

**TEACHUCOMP, INC.®**

*...it's all about you*

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# MASTERING INTRODUCTORY JAVASCRIPT MADE EASY™ v.1.0

## Copyright:

Copyright © 2014 by TeachUcomp, Inc. All rights reserved. This publication, or any part thereof, may not be reproduced or stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, recording, photocopying, or otherwise, without the express written permission of TeachUcomp, Inc.

For PDF manuals, TeachUcomp, Inc. allows the owner of the PDF manual to make up to 2 additional copies of the PDF manual that the owner may place on up to 2 additional non-shared computer hard drives for ease of use when using the accompanying video lessons. TeachUcomp, Inc. also grants unlimited personal printing rights to the owner, strictly limited to the purposes of personal or private education or research.

The unauthorized reproduction or distribution of this copyrighted work is illegal. Criminal copyright infringement, including infringement without monetary gain, is investigated by the FBI and is punishable by up to five years in federal prison and a fine of \$250,000.

## Trademark Acknowledgements:

Apple, Macromedia, Dreamweaver, CoffeeCup Software, eBay, Google, Intuit, Quicken, QuickBooks, QuickBooks Pro, QuickBooks Premier, Turbo Tax, EasyStep, QuickReports, and QuickZoom are registered trademarks of Intuit, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows Me, Windows XP, Windows 7, Windows 8, Microsoft Word 97, Microsoft Word 2000, Microsoft Word XP, Microsoft Word 2003, Microsoft Word 2007, Microsoft Word 2013, Microsoft Excel 97, Microsoft Excel 2000, Microsoft Excel XP, Microsoft Excel 2003, Microsoft Excel 2007, Microsoft Excel 2013, and Outlook are registered trademarks of Microsoft Corporation. Other brand names and product names are trademarks or registered trademarks of their respective holders.

## Disclaimer:

While every precaution has been made in the production of this book, TeachUcomp, Inc. assumes no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein. These training materials are provided without any warranty whatsoever, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. All names of persons or companies in this manual are fictional, unless otherwise noted.

**TEACHUCOMP, INC.**

Phone: (877) 925-8080

Web: <http://www.teachucomp.com>

# INTRODUCTION AND OVERVIEW

---

Welcome to TeachUcomp, Inc.'s Mastering Introductory JavaScript Made Easy™ v.1.0 Course. This course introduces the student to the JavaScript programming language used to change, add interest to, and automate webpages.

The purpose of this course is to educate the student in the basic language skills necessary to use JavaScript. There are many resources available on the Internet that allow you to download code and place it into your HTML document or JavaScript code file. However, it is important to first understand the underlying language and components of JavaScript to be able to alter the code to work in your desired application, and easily troubleshoot any errors that may occur.

Whether you are looking to add interactivity to your website, control how a browser acts or alter your HTML document's content, it is a good idea to have a firm grasp of the basics of JavaScript.

We will begin this course by discussing the basic components and structure of JavaScript as well as learning the terminology. Then, we'll advance through topics to cover some more advanced concepts and uses for JavaScript.

# TABLE OF CONTENTS

<u>Chapters/Lessons:</u>	<u>Page(s):</u>	<u>Chapters/Lessons:</u>	<u>Page(s):</u>
<b>CHAPTER 1- Getting Acquainted with JavaScript</b>	<b>8</b>	<b>CHAPTER 8- JavaScript Operators</b>	<b>46</b>
1.1- Introduction to JavaScript	9	8.1- Arithmetic Operators	47
1.2- JavaScript vs. Java	9	8.2- Assignment Operators	47
1.3- The <SCRIPT>...</SCRIPT> Tag	9	8.3- Adding Strings and Numbers	48
1.4- External JavaScript	10	8.4- Comparison Operators	49
1.5- Uses for JavaScript	10	8.5- Logical Operators	50
<i>Getting Acquainted with JavaScript- Actions</i>	11	8.6- Conditional Operators	51
<i>Getting Acquainted with JavaScript- Exercises</i>	12	<i>JavaScript Operators- Actions</i>	52
		<i>JavaScript Operators- Exercises</i>	53
<b>CHAPTER 2- The Makeup of JavaScript</b>	<b>13</b>	<b>CHAPTER 9- JavaScript Conditions</b>	<b>54</b>
2.1- JavaScript Statements	14	9.1- If Statements	55
2.2- Code and Code Blocks	14	9.2- The Switch Statement	56
2.3- Whitespace	15	<i>JavaScript Conditions- Actions</i>	57
2.4- Case Sensitivity	15	<i>JavaScript Conditions- Exercises</i>	58-59
2.5- Breaking Up a Line of Code	15		
<i>The Makeup of JavaScript- Actions</i>	16	<b>CHAPTER 10- JavaScript Loops</b>	<b>60</b>
<i>The Makeup of JavaScript-Exercises</i>	17	10.1- The FOR Loop	61
<b>CHAPTER 3- JavaScript Comments</b>	<b>18</b>	10.2- The FOR...IN Loop	62
3.1- Single Line Comments	19	10.3- The WHILE Loop	63
3.2- Multi-Line Comments	19	10.4- The DO...WHILE Loop	64
3.3- End of Line Comments	20	<i>JavaScript Loops- Actions</i>	65
3.4- Using Comments to Stop Execution	20	<i>JavaScript Loops- Exercises</i>	66
<i>JavaScript Comments- Actions</i>	21	<b>CHAPTER 11- JavaScript Break and Continue</b>	<b>67</b>
<i>JavaScript Comments- Exercises</i>	22	11.1- The Break Statement	68
<b>CHAPTER 4- JavaScript Variables</b>	<b>23</b>	11.2- The Continue Statement	69
4.1- What are JavaScript Variables?	24	11.3- JavaScript Labels	70
4.2- Syntax for Text and Numerical Values	24	<i>JavaScript Break and Continue- Actions</i>	71
4.3- Creating (Declaring) Variables	24	<i>JavaScript Break and Continue- Exercises</i>	72
4.4- Re-Declaring Variables	24		
4.5- Undefined Value	25	<b>CHAPTER 12- JavaScript Errors</b>	<b>73</b>
4.6- Using One Statement for Multiple Variables	25	12.1- The Try...Catch Statement	74
4.7- Local Variables and Global Variables	26	12.2- The Throw Statement	75
<i>JavaScript Variables- Actions</i>	27	<i>JavaScript Errors- Actions</i>	76
<i>JavaScript Variables- Exercises</i>	28	<i>JavaScript Errors- Exercises</i>	77
<b>CHAPTER 5- Exploring JavaScript Data Types</b>	<b>29</b>	<b>CHAPTER 13- JavaScript Form Validation</b>	<b>78</b>
5.1- Dynamic Data Types in JavaScript	30	13.1- Form Validation	79
5.2- Null	30	13.2- EMail Validation	80
5.3- Number	30	<i>JavaScript Form Validation- Actions</i>	81
5.4- String	30	<i>JavaScript Form Validation- Exercises</i>	82
5.5- Boolean	31		
5.6- Array	31	<b>CHAPTER 14- JavaScript RegExp Object</b>	<b>83</b>
5.7- Object	32	14.1- RegExp Definition and Modifiers	84-85
<i>Exploring JavaScript Data Types- Actions</i>	33	14.2- RegExp Special Characters	86-87
<i>Exploring JavaScript Data Types- Exercises</i>	34	14.3- RegExp Methods	88
<b>CHAPTER 6- JavaScript Objects</b>	<b>35</b>	<i>JavaScript RegExp Object- Actions</i>	89
6.1- Creating Objects	36	<i>JavaScript RegExp Object- Exercises</i>	90
6.2- Accessing Object Properties	36		
6.3- Accessing Object Methods	37	<b>CHAPTER 15- JavaScript Hoisting</b>	<b>91</b>
<i>JavaScript Objects- Actions</i>	38	15.1- Declarations	92
<i>JavaScript Objects- Exercises</i>	39	15.2- Initializations	93
		<i>JavaScript Hoisting- Actions</i>	94
<b>CHAPTER 7- JavaScript Functions</b>	<b>40</b>	<i>JavaScript Hoisting- Exercises</i>	95
7.1- JavaScript Function Definition and Syntax	41		
7.2- Functions with a Return Value	42	<b>CHAPTER 16- JavaScript USE STRICT Directive</b>	<b>96</b>
7.3- Calling a Function with Arguments	42	16.1- What is the USE STRICT Directive and Why Use It?	97
7.4- Assigning Values to Undeclared Variables	43	16.2- What's Not Allowed in STRICT Mode?	98-101
<i>JavaScript Functions- Actions</i>	44	<i>JavaScript USE STRICT Directive- Actions</i>	102
<i>JavaScript Functions- Exercises</i>	45	<i>JavaScript USE STRICT Directive- Exercises</i>	103

# TABLE OF CONTENTS

<u>Chapters/Lessons:</u>	<u>Page(s):</u>	<u>Chapters/Lessons:</u>	<u>Page(s):</u>
<b>CHAPTER 17- JavaScript HTML DOM</b>	<b>104</b>	<b>CHAPTER 25- Window Navigator Object</b>	<b>155</b>
17.1- What is HTML DOM?	105	25.1- What Does the Window Navigator Object Do?	156
17.2- HTML DOM Methods and Properties	106	<i>Window Navigator Object- Actions</i>	157
17.3- HTML DOM Document	106	<i>Window Navigator Object- Exercises</i>	158
17.4- Finding HTML Elements	107		
17.5- Changing the Output Stream	108	<b>CHAPTER 26- JavaScript Popup Boxes</b>	<b>159</b>
17.6- Changing the Value of an Attribute	109	26.1- The Alert Box	160
17.7- Changing CSS	110	26.2- The Confirm Box	161
<i>JavaScript HTML DOM- Actions</i>	111	26.3- The Prompt Box	162
<i>JavaScript HTML DOM- Exercises</i>	112	<i>JavaScript Popup Boxes- Actions</i>	163
		<i>JavaScript Popup Boxes- Exercises</i>	164
<b>CHAPTER 18- HTML DOM Events</b>	<b>113</b>		
18.1- Using Events	114	<b>CHAPTER 27- JavaScript Timing Events</b>	<b>165</b>
18.2- The ONCHANGE Event	115	27.1- What are JavaScript Timing Events?	166
<i>HTML DOM Events- Actions</i>	116	27.2- SETINTERVAL() and CLEARINTERVAL() Methods	166-167
<i>HTML DOM Events- Exercises</i>	117	27.3- SETTIMEOUT() and CLEARTIMEOUT() Methods	168
		27.4- Creating a Clock	169
<b>CHAPTER 19- HTML DOM Navigation</b>	<b>118</b>	<i>JavaScript Timing Events- Actions</i>	170
19.1- DOM Nodes	119	<i>JavaScript Timing Events- Exercises</i>	171
19.2- Node Relationships	120		
19.3- Child Nodes and Values	121	<b>CHAPTER 28- JavaScript Cookies</b>	<b>172</b>
19.4- NODE... Properties	122	28.1- What are Cookies?	173
19.5- HTML DOM Nodelist	123-124	28.2- Working with Cookies	173-178
19.6- Root Nodes	125	<i>JavaScript Cookies- Actions</i>	179
<i>HTML DOM Navigation- Actions</i>	126	<i>JavaScript Cookies- Exercises</i>	180-181
<i>HTML DOM Navigation- Exercises</i>	127		
<b>CHAPTER 20- Adding and Removing DOM Nodes</b>	<b>128</b>	<b>CHAPTER 29- The JavaScript Console Object</b>	<b>182</b>
20.1- Creating New HTML Elements (Nodes)	129-130	29.1- The Console Object	183
20.2- Removing Existing HTML Elements (Nodes)	131	29.2- Inline Grouping	184
20.3- Replacing HTML Elements (Nodes)	132	29.3- Timers	185
<i>Adding and Removing HTML DOM Nodes- Actions</i>	133	29.4- String Substitution	185
<i>Adding and Removing HTML DOM Nodes- Exercises</i>	134	<i>The JavaScript Console Object- Actions</i>	186
		<i>The JavaScript Console Object- Exercises</i>	187
<b>CHAPTER 21- JavaScript Browser Object Model (BOM)</b>	<b>135</b>		
21.1- What is the Browser Object Model (BOM)?	136	<b>CHAPTER 30- Advanced JavaScript Objects</b>	<b>188</b>
21.2- The Window Object	137	30.1- The Object Literal and the Keyword New	189-190
21.3- Window Size Properties	138	30.2- Using an Object Constructor	190-191
21.4- Other Window Methods and Properties	139	30.3- JavaScript Prototype	191-192
<i>JavaScript Browser Object Model (BOM)- Actions</i>	140	30.4- Mutable Objects and Immutable Primitive Values	193-194
<i>JavaScript Browser Object Model (BOM)- Exercises</i>	141	30.5- JavaScript Properties	194-195
		30.6- Adding New Properties and Deleting Properties	195-196
<b>CHAPTER 22- Window Screen Object</b>	<b>142</b>	<i>Advanced JavaScript Objects- Actions</i>	197
22.1- What Does the Window Screen Object Do?	143	<i>Advanced JavaScript Objects- Exercises</i>	198
22.2- Window Screen Object Properties	143		
<i>Window Screen Object- Actions</i>	144	<b>CHAPTER 31- Number Object</b>	<b>199</b>
<i>Window Screen Object- Exercises</i>	145	31.1- What is a Number Object?	200
		31.2- Hexadecimal Numbers	200
<b>CHAPTER 23- Window Location Object</b>	<b>146</b>	31.3- NaN – Not a Number	201
23.1- What Does the Window Location Object Do?	147	31.4- Infinity	202
23.2- Window Location HREF Property	147	<i>Number Object- Actions</i>	203
23.3- Window Location PATHNAME Property	148	<i>Number Object- Exercises</i>	204
23.4- Window Location ASSIGN() Method	148		
<i>Window Location Object- Actions</i>	149	<b>CHAPTER 32- String Object</b>	<b>205</b>
<i>Window Location Object- Exercises</i>	150	32.1- Using the String Object	206
		32.2- String Properties and Methods	207-208
<b>CHAPTER 24- Window History Object</b>	<b>151</b>	32.3- Special Characters	209
24.1- What Does the Window History Object Do?	152	<i>String Object- Actions</i>	210
24.2- Window History Back and Forward Methods	152	<i>String Object- Exercises</i>	211
<i>Window History Object- Actions</i>	153		
<i>Window History Object- Exercises</i>	154		

# TABLE OF CONTENTS

<u>Chapters/Lessons:</u>	<u>Page(s):</u>
<b>CHAPTER 33- Date Object</b>	<b>212</b>
33.1- The Date Object	213
33.2- Set and Compare Dates	214
33.3- Convert the Date to a String	215
<i>Date Object- Actions</i>	216
<i>Date Object- Exercises</i>	217
<b>CHAPTER 34- Array Object</b>	<b>218</b>
34.1- Create and Access an Array	219
34.2- Joining Arrays	219-220
34.3- Working with Arrays	220-222
<i>Array Object- Actions</i>	223
<i>Array Object- Exercises</i>	224
<b>CHAPTER 35- Math Object</b>	<b>225</b>
35.1- The Math Object and Mathematical Constants	226
35.2- Math Object Methods	227
<i>Math Object- Actions</i>	228
<i>Math Object- Exercises</i>	229
<b>CHAPTER 36- JavaScript Libraries (Frameworks)</b>	<b>230</b>
36.1- JavaScript Libraries or Frameworks	231
36.2- Testing jQuery	232-233
<i>JavaScript Libraries (Frameworks)- Actions</i>	234
<i>JavaScript Libraries (Frameworks)- Exercises</i>	235
<b>Appendix</b>	
JavaScript Reserved Words	236
RegExp Special Characters	237-239
RegExp Methods	240
HTML Document Object Methods	241-243
HTML Element Object Methods	244-247
HTML Attribute Object Methods	248
HTML Style Object Properties	249-257
HTML DOM Events	258-259
HTML DOM Event Object	260
HTML DOM EventTarget Object	261
HTML MouseEvent/KeyboardEvent Object	262
HTML DOM Navigation NodeNames	263
HTML DOM Nodes	264
HTML DOM Navigation NodeType Values	265
JavaScript Window Object	266-267
JavaScript Screen Object	268
JavaScript Location Object	269
JavaScript History Object	270
JavaScript Navigator Object	271
JavaScript Console Object	272-273
Number Object	274
String Object	275-276
Date Object	277-279
Array Object	280
Math Object	281-283

# ABOUT THIS MANUAL

## Menu Bar or Ribbon:

When menu items from the menu bar or ribbon are referenced, the main menu title will be displayed, followed by a “|”, followed by the menu item.

Example: Edit| Copy.

## Keyboard Shortcuts:

When keyboard shortcuts are referenced, the keyboard combination will be displayed as the first key which is held down, followed by a “+”, followed by the second key which is pressed and released quickly.

Example: CTRL+A

## JavaScript Code:

In this manual when a specific code is referenced it will appear in **bold text**. This is to allow the code to stand out from the instruction text.

Example: **var person="Thomas"**

# CHAPTER 1-

## GETTING ACQUAINTED WITH JAVASCRIPT

---

1.1- INTRODUCTION TO JAVASCRIPT

1.2- JAVASCRIPT VS. JAVA

1.3- THE `<SCRIPT> ... </SCRIPT>` TAG

1.4- EXTERNAL JAVASCRIPT

1.5- USES FOR JAVASCRIPT

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)



# GETTING ACQUAINTED WITH JAVASCRIPT

## 1.1- Introduction to JavaScript:

JavaScript is considered a dynamic programming language. It was first released with Netscape Navigator in 1995. Originally called *LiveScript*, the name was changed when Netscape Navigator added support for the machine language Java.

It quickly gained success and Microsoft introduced JavaScript in the release of Internet Explorer Version 3.0, in 1996. Since its introduction in 1995, JavaScript has become one of the most popular programming languages for the web.

Writing functions that are embedded in HTML pages is the most common use of JavaScript today. But, it can also be used to animate page elements, validate inputted data and many other functions that enhance your users' experience. JavaScript runs locally on your users' browser, making it possible for quick reaction times. This also allows for a much more responsive experience on the web.

JavaScript is currently the only programming language that is supported by most of the popular browsers used worldwide. Because of this it is the target language for many different frameworks in other languages. This allows for greater expansion and more widespread use as a programming language. JavaScript is currently a trademark of the Oracle Corporation and is licensed for use by current entities such as the Mozilla Foundation, creators of the popular browser Firefox.

Before writing JavaScript code, it is a good idea to have a working knowledge of HTML (HyperText Markup Language) and Cascading Style Sheets (CSS).

## 1.2- JavaScript vs. Java:

Although the two programming languages Java and JavaScript have similar names, they are completely different in function. Java was created by Sun Microsystems and is used as a general programming language. JavaScript was created by the people who made the Web browser Netscape Navigator and is used to animate and add user interactions to webpages.

Both languages are Object Oriented Programming (OOP) languages and can have similar programming structures, but that is where the similarities end. Java is a much more complex language and is designed to function on its own. JavaScript has a much smaller set of commands and cannot stand on its own. JavaScript must be inserted into, or linked to, an HTML document to function properly.

Another distinct difference is the way the languages are read. JavaScript is composed using the English language and is implemented the way it is input. Java is written in English and then compiled by another computer program and rewritten into a machine language. The machine language is then put into use by the computers it is installed on and runs Java.

## 1.3- The <SCRIPT>...</SCRIPT> Tag:

The `<script>...</script>` tag is used to insert JavaScript code into your HTML document. It tells the web browser where your JavaScript starts and the lines of code in between the tags contain all your coding.

The most common area to place `<script>` tags is between the `<head>...</head>` tags and between the `<body>...</body>` tags. Although it is commonly placed within the mentioned tags, your JavaScript coding can be placed anywhere in your HTML document.

# GETTING ACQUAINTED WITH JAVASCRIPT

## 1.4- External JavaScript:

The most effective use of JavaScript is to create an external **.js** file. Using an external JavaScript file allows your browser to preload, or cache, all the JavaScript code for your whole website. This minimizes your users wait time as pages on your website will load faster, making for a better user experience.

Much like Cascading Style Sheets, you create your JavaScript code using a simple text editor like, Microsoft Notepad or Apple TextEdit, and saving the document with the **.js** file extension. The correct syntax when linking to an external JavaScript document is:

```
<script src="testscript.js"></script>
```

Where "testscript" is replaced with the name you choose for your JavaScript file. It is a good idea to keep this name concise and easy to reference to the webpage or website it is attached to.

## 1.5- Uses for JavaScript:

The uses for JavaScript are almost endless. They range from validating the information users input in a form to embedded games on your webpage. The most widely recognized use of JavaScript is probably the Google Search Engine homepage. JavaScript has allowed them to remove extraneous items from the page to leave a clean, minimalist look.

Google also implements JavaScript when running their analytics algorithms. This allows you to keep track of how your users interact with your website. It can tell you what they read on your page, how long they stayed on your page and just about anything you would want to learn about how your webpages are used.

# ACTIONS-

## GETTING ACQUAINTED WITH JAVASCRIPT

### THE <SCRIPT>...</SCRIPT> TAGS:

1. In the HTML document you would like to add JavaScript to, type: <script>
2. Enter the whole of your JavaScript code
3. To end the JavaScript portion of your HTML document, type: </script>

### EXTERNAL JAVASCRIPT:

1. In the text editor of your choice, enter the JavaScript code for your webpage.
2. Select "File| Save As..." from the Menu Bar or Ribbon.
3. In the Dialog Box, navigate to the correct folder where you want to save your JavaScript document.
4. Use the drop-down under "Save as type" and select ".txt."
5. In the "File Name" box, enter a descriptive name for your JavaScript document and end it with the .js file extension.
6. Click "Save".

# EXERCISES-

## GETTING ACQUAINTED WITH JAVASCRIPT

### Purpose:

1. Add the JavaScript `<script>...</script>` tag to an HTML page.

### Exercises:

1. Open a text or HTML editor and start a new page.
2. Open your HTML editor and begin a new page.
3. Type: `<!DOCTYPE html>`
4. Press "Enter".
5. Type: `<html>`
6. Press "Enter".
7. Type: `<head>`
8. Press "Enter".
9. Type: `<script>`
10. Press "Enter".
11. Type: `</script>`
12. Press "Enter".
13. Type: `</head>`
14. Press "Enter".
15. Type: `<body>`
16. Press "Enter".
17. Type: `</body>`
18. Press "Enter".
19. Type: `</html>`
20. Press "Enter".
21. In your text or HTML editor, Select "File | Save As..." from the Menu Bar or Ribbon.
22. In the Dialog Box, navigate to the correct folder where you want to save your document.
23. Use the drop-down under "Save as type" and select ".txt".
24. In the "File name" box, enter "my-practice-page" with a .html or .htm extension.
25. Click "Save".

# CHAPTER 2-

## THE MAKEUP OF JAVASCRIPT

---

**2.1- JAVASCRIPT STATEMENTS**

**2.2- CODE AND CODE BLOCKS**

**2.3- WHITESPACE**

**2.4- CASE SENSITIVITY**

**2.5- BREAKING UP A LINE OF CODE**

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# THE MAKEUP OF JAVASCRIPT

## 2.1- JavaScript Statements:

JavaScript statements are the commands to the browser to execute the code you input. Much like CSS, the semicolon (;) is used to separate JavaScript statements and allows for many statements to be typed on a single line. Usually there is also a semicolon at the end of executable statements. Unlike CSS, you do **NOT** have to end an executable JavaScript statement with a semicolon, although it is good practice to do so. The following example is a JavaScript statement that tells the browser to write "Basic JavaScript" in any HTML element with the ID attribute "test":

```
document.getElementById("test").innerHTML="Basic JavaScript";
```

## 2.2- Code and Code Blocks:

JavaScript code is simply a sequence of JavaScript statements. Each statement is executed by the browser in the order they are entered into your HTML document or external .js file.

JavaScript statements can be grouped together in blocks. Code blocks start with a left curly bracket ( { ) and end with a right curly bracket ( } ). The purpose of blocks is to allow the JavaScript code to execute together. Code blocks are useful when using JavaScript functions. The following is a good example of using code blocks in a JavaScript function.

<b>Start Tag:</b>	<script>
<b>End Tag:</b>	</script>
<b>Code Block Start "Tag":</b>	{
<b>Code Block End "Tag":</b>	}
<b>Example:</b>	<pre>&lt;script&gt; function myFunction() { document.getElementById("line").innerHTML="Are you sure?"; document.getElementById("test").innerHTML="Yes, I'm positive."; } &lt;/script&gt;</pre>
<b>Explanation:</b>	Runs the JavaScript function " <b>myFunction()</b> " that will change, or manipulate, two separate HTML elements simultaneously.

# THE MAKEUP OF JAVASCRIPT

## 2.3- Whitespace:

In JavaScript, extraneous whitespace is ignored. This means that if you have extra space between any of the terms in your code, JavaScript will ignore it and not cause the line of code to fail. For example:

```
var person="Thomas";
```

Is considered the same as:

```
var person = "Thomas" ;
```

## 2.4- Case Sensitivity:

JavaScript is case sensitive. Cases in corresponding functions and variables must match exactly or they will not be rendered properly. Most JavaScript code is written in camelCase, which means compound words written with the first letter in lowercase and subsequent words with their first letter capitalized. While the first letter, in the first word, of a camelCase word can be either lowercase or capitalized, in most computer programming languages it is common practice for it to be lowercase. For example:

```
getElementById
```

Is not the same as:

```
getElementbyId
```

## 2.5- Breaking Up a Line of Code:

JavaScript allows you to break up a line of code within a text string with a backslash (\). You cannot break up a line of code anywhere else in the statement. For example:

```
document.write("Stop, \nDon't Stop");
```

Is correct.

*But*

```
document.write(\n("Stop, Don't Stop");
```

Is not correct and will not be rendered properly, this code will throw an error message or will be ignored entirely, by the browser.

# ACTIONS-

## THE MAKEUP OF JAVASCRIPT

---

### CODE AND CODE BLOCKS:

1. In the HTML document you want to add JavaScript code to, type: `<script>`
  2. On a new line, or lines, type the JavaScript code you want to add.
  3. If you are adding a function, or code block, enter the code between two curly brackets `{...}`.
  4. When you are finished entering your code, close the JavaScript portion of your document by typing: `</script>`
  5. Everything you place between the `<script>...</script>` tags is your JavaScript code.
- 

### BREAKING UP A LINE OF CODE:

1. In the text string, of a line of code you want to break up, place a backslash `\`

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)



# EXERCISES-

## THE MAKEUP OF JAVASCRIPT

---

### Purpose:

1. Covered at the end of a following chapter.
- 

### Exercises:

1. Covered at the end of a following chapter.

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# CHAPTER 3-

## JAVASCRIPT COMMENTS

---

**3.1- SINGLE LINE COMMENTS**

**3.2- MULTI-LINE COMMENTS**

**3.3- END OF LINE COMMENTS**

**3.4- USING COMMENTS TO STOP EXECUTION**

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# JAVASCRIPT COMMENTS

## 3.1- Single Line Comments:

Comments in JavaScript are primarily used to explain what the code is for or to make it more readable. Single line comments start with a double forward slash (`//`), but do not have an end tag. Any text that is written after (`//`) will be completely ignored by JavaScript and will not be executed or displayed by the browser. JavaScript comments can also be used to stop the execution of code, which we will cover in a later lesson.

<b>Start “Tag”:</b>	<code>//</code>
<b>End “Tag”:</b>	None
<b>Example:</b>	<code>// Write to element with ID of “line”. document.getElementById(“line”).innerHTML= “Are you sure?”;</code>
<b>Explanation:</b>	Labels JavaScript code so anyone viewing will know the statement is to write to an element with an ID of “line”.

## 3.2- Multi-line Comments:

Multi-line comments in JavaScript start with a forward slash and an asterisk (`/*`) and end with an asterisk and a forward slash (`*/`). This allows for breaking your comments up into many lines, making it more readable for later editing. Just like single line comments, multi-line comments are ignored by JavaScript and will not be displayed by browsers.

<b>Start “Tag”:</b>	<code>/*</code>
<b>End “Tag”:</b>	<code>*/</code>
<b>Example:</b>	<code>/* This is an Example of a Multi-line comment. */</code>
<b>Explanation:</b>	Allows you to write comments across multiple lines that will be ignored by JavaScript and web browsers.

# JAVASCRIPT COMMENTS

## 3.3- End of Line Comments:

You can also add comments to the end of a line of JavaScript code. These are considered single line comments and begin with a double forward slash (`//`). Since anything after the `//` is ignored, your comments will not be displayed by the browser.

<b>Start "Tag":</b>	<code>//</code>
<b>End "Tag":</b>	None
<b>Example:</b>	<code>var person="Thomas" // Sets variable to value of Thomas.</code>
<b>Explanation:</b>	Labels JavaScript code so anyone viewing will know the variable has a value of Thomas.

## 3.4- Using Comments to Stop Execution:

You can use the single line comment "tag" (`//`), or the multi-line comment "tags" (`/*...*/`) to stop the execution of a line, or multiple lines, of code. This works because JavaScript ignores any text entered after the comment "tags".

This is useful when you want to keep the code, but temporarily disable it. It is also useful when you are trying to debug an element or function in JavaScript. To re-activate a line of code that has been deactivated simply remove the comment "tags" and the code will function as normal.

<b>Single Line Comment Start "Tag":</b>	<code>//</code>
<b>Single Line Comment End "Tag":</b>	None
<b>Multi-Line Comment Start "Tag":</b>	<code>/*</code>
<b>Multi-Line Comment End "Tag":</b>	<code>*/</code>
<b>Example:</b>	<code>// document.getElementById("line").innerHTML="Are you sure?";  /* document.getElementById("head1").innerHTML="Heading One"; document.getElementById("para").innerHTML="First Paragraph"; */</code>
<b>Explanation:</b>	Stops the lines of code from functioning as JavaScript as they are considered comments and will be ignored until the "tags" are removed.

# ACTIONS- JAVASCRIPT COMMENTS

## SINGLE LINE COMMENTS:

1. To add a single line comment, type: //
2. On the same line enter the information you want as your comment.

## MULTI-LINE COMMENTS:

1. On the line you want your comment to start, type: /\*
2. On a new line start the comment you want to enter
3. When your comment is complete, type: \*/

## END OF LINE COMMENTS:

1. On the line of code you want to enter a comment on, after the code, type: //
2. On the same line enter the comment

## USING COMMENTS TO STOP EXECUTION:

1. On the line of code you no longer want to execute, before the code, type: //
2. If you have multiple lines of code you would like to stop execution of, on the line before the first line of code, type: /\*
3. On the line after the last line of code, type: \*/

# EXERCISES- JAVASCRIPT COMMENTS

---

## Purpose:

1. To add comments to simple JavaScript code.
- 

## Exercises:

1. Open the HTML document my-practice-page.html, created in a previous chapter.
2. On the line with the <script> tag, press “Enter” to start a new blank line.
3. Type: document.write(“My first JavaScript”);
4. Press “Enter”.
5. Type: //This will write the parameter "My first JavaScript" to your HTML page
6. Press “Enter”.
7. In your text or HTML editor, Select “File| Save” from the Menu Bar or Ribbon, or use the keyboard shortcut CTRL+S to save your document

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# **CHAPTER 4-**

## **JAVASCRIPT VARIABLES**

---

**4.1- WHAT ARE JAVASCRIPT VARIABLES?**

**4.2- SYNTAX FOR TEXT AND NUMERICAL VALUES**

**4.3- CREATING (DECLARING) VARIABLES**

**4.4- RE-DECLARING VARIABLES**

**4.5- UNDEFINED VALUE**

**4.6- USING ONE STATEMENT FOR MULTIPLE VARIABLES**

**4.7- LOCAL VARIABLES AND GLOBAL VARIABLES**

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# JAVASCRIPT VARIABLES

## 4.1- What are JavaScript Variables?:

Much like algebra, JavaScript uses letters as containers for storing information, for example: **var x=7;**. In the previous example, the letter "x" is assigned a numerical value of 7, declaring that any variables with the letter "x" will have a value of 7.

JavaScript variables can hold expressions as well as single values. This allows JavaScript to be able to perform arithmetic by using operators like = and +, for example: **var c=a+b;**

JavaScript variables can have single letter names, like **x** or descriptive names, like **sum** or **totalvolume**. It is important to note the JavaScript variables **must** start with a letter. Just like JavaScript statements, variables are case sensitive, where y and Y are not the same.

## 4.2- Syntax for Text and Numerical Values:

JavaScript variables can have either a numerical or textual value. The syntax for each variable is different. When assigning a numerical value to a variable you only need to input the number: **var a=7;**

When assigning a textual value to a variable you must surround your value with either single or double quotation marks: **var a="Yes";**. A textual value for a variable is considered a "string". If you place quotation marks around a numerical value it will be treated as text and will not function properly if there is a later mathematical function in the same statement that calls on that variable.

## 4.3- Creating (Declaring) Variables:

When creating variables using JavaScript, you "declare" the variable by giving it a value. JavaScript variables are declared using the keyword **var**. For ease of reading within your code, you can declare a variable and on the next line assign a value to that variable. You can also assign a value on the same line as you declare a variable. This allows for smaller file size and faster loading times. It is good practice to declare all your variables at the beginning of your code. Doing so makes for easier editing and debugging. For example, the following lines of code will have the same end result:

```
var person;  
person="John Smith";  
Or  
var person="John Smith";
```

## 4.4- Re-Declaring Variables:

If you re-declare a JavaScript variable, that has previously been given a value, it does not lose its value. For example, after the execution of the following lines of code, the variable "person" will keep the value of John Smith.

```
var person="John Smith";  
var person;
```



# JAVASCRIPT VARIABLES

---

## 4.5- Undefined Value:

In JavaScript a variable that is not given a specific value is considered undefined. The value of a variable is something that has to be calculated or something that may be provided later, like user inputted data. For example:

```
var person;
```

Has an *undefined* value after the execution of the statement because no definite value was given to "person".

---

## 4.6- Using One Statement for Multiple Variables:

When programming code with JavaScript, you can have many variables in one statement. This is achieved by starting the statement with **var** and listing your variables all separated by commas. For example:

```
var person="John Smith", weight=165, eyecolor="blue";
```

A single statement with multiple variables can also be expressed on multiple lines. For example, the following code is the same as the previous code.

```
var person="John Smith",  
weight=165,  
eyecolor="blue";
```

# JAVASCRIPT VARIABLES

## 4.7- Local Variables and Global Variables:

In JavaScript a "local" variable is a variable that is contained within a function. It will not be recognized by any other function in your full JavaScript coding. Because of this, you can have local variables in other functions that have the same value. When a function is complete any local variables are deleted.

If a variable has been declared that is **NOT** contained within a function, it is considered a "global" variable. This allows all scripts and functions on a webpage to access it.

If you have a global variable with the same name as a local variable, the local variable will be accessed first and the value will supersede the global variable value, until the function has been executed. After execution the value will be assigned with the global variable value.

The lifetime of variables depends on whether it is local or global. A local variable is deleted when the function is complete. A global variable is deleted when you close the page.

<b>Local Variable:</b>	<pre>function foo () {   var x = 9; }</pre>
<b>Explanation:</b>	The variable <b>y</b> with a value of <b>9</b> is considered to be "local" as it is contained within a specific function, in this instance " <b>foo</b> ". When the function has been executed the variable will be deleted and irrelevant to any future functions or calculations.
<b>Global Variable:</b>	<pre>var x=7;</pre>
<b>Explanation:</b>	The variable <b>x</b> with a value of <b>7</b> is considered to be "global" as it is not contained within a specific function.

# ACTIONS- JAVASCRIPT VARIABLES

## CREATING (DECLARING) VARIABLES:

1. To create a text string variable, Type: `var x = "y";`  
Where "x" is the name of your variable and "y" is the value in quotation marks.
2. To create a numerical variable, Type: `var x = y;`  
Where "x" is the name of your variable and "y" is the numerical value.

## UNDEFINED VALUE:

1. To create a variable with an undefined value, type: `var x;`  
Where "x" is the name of the variable you are creating.

## USING ONE STATEMENT FOR MULTIPLE VARIABLES:

1. Type: `var a="b", c="d", e=f;`  
**OR**  
`var a="b",`  
`c="d",`  
`e=f;`

Where "a", "c" and "e" are the names of your variables, and "b", "d" and "f" are the values.

## LOCAL AND GLOBAL VARIABLES:

1. To create a local variable, type: `function a() {var x="y";}`  
Where "a" is the name of your function, "x" is the name of your variable and "y" is the value of your variable.
2. To create a global variable, type `var a="b";`  
Where "a" is the name of the variable and "b" is the value of your variable.

# EXERCISES- JAVASCRIPT VARIABLES

---

## Purpose:

1. To add a variable to your JavaScript code.
- 

## Exercises:

1. Open the HTML document my-practice-page.html, created in a previous chapter.
2. At the end of the last line, before the </script> tag, press “Enter”.
3. Type: `var person=“John Smith”, weight=180, eyecolor=“Blue”;`
4. Press “Enter”.
5. In your text or HTML editor, Select “File| Save” from the Menu Bar or Ribbon, or use the keyboard shortcut CTRL+S to save your document

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# CHAPTER 5-

## EXPLORING JAVASCRIPT DATA TYPES

---

**5.1- DYNAMIC DATA TYPES IN JAVASCRIPT**

**5.2- NULL**

**5.3- NUMBER**

**5.4- STRING**

**5.5- BOOLEAN**

**5.6- ARRAY**

**5.7- OBJECT**

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)

# EXPLORING JAVASCRIPT DATA TYPES

## 5.1- Dynamic Data Types in JavaScript:

Variables in JavaScript can have different data types. This allows variables to be considered "dynamic". There are seven different data types that can be attributed to variables in JavaScript. They are; undefined, null, number, string, Boolean, array and object. Each data type is covered in the following lessons.

## 5.2- Null:

You can attribute a *null* value to variables to empty them. A null value data type is different from an undefined data type, because a null value data type has been labeled as "null" and not just left empty. When a variable has no definition it is *undefined*, meaning it has no value, no type and has never been referenced before. Uninitialized variables, missing parameters and unknown variables are all considered *undefined*. A variable with the definition of *null* means the property exists and is given the value of "no value" or *null* to be used as a parameter in a function.

## 5.3- Number:

In JavaScript there is only one type of number. Numbers can be written with or without decimals and extra large or extra small numbers can be expressed using scientific or exponential notations.

<b>Without Decimals:</b>	<code>var x=7;</code>	Will display the number 7.
<b>With Decimals:</b>	<code>var y=9.00;</code>	Will display the number 9.
<b>Scientific Notation:</b>	<code>var a=7e5;</code>	Will display the number 700000 or 7 to the power of 10 times 5.

## 5.4- String:

A JavaScript String object is simply any variable ascribed a value in text form, for example:

```
var x="John Smith";
```

Variables with string values can be surrounded by either double or single quotation marks. As mentioned previously any numerical values contained in quotation marks will be considered string objects and will be considered as if they were simply text to JavaScript.

Your string object can have quotation marks inside your containing quotation marks, as long as they are different. The following example will be displayed as 'John Smith':

```
var x=" 'John Smith' ";
```

# EXPLORING JAVASCRIPT DATA TYPES

## 5.5 Boolean:

Boolean data types have only two values TRUE and FALSE. Any variable labeled with either TRUE or FALSE is considered a Boolean data type. One of the main uses for Boolean data types is conditional testing, which is covered in a later chapter. An example of a Boolean data type:

```
var x=true;
```

It is important to note that Boolean data types do **NOT** have quotations marks around their values, to do so would make them string data types.

## 5.6- Array:

In JavaScript, arrays are actually objects, not a separate “data type”. An array is simply a single variable with many different values. You access the values by referencing index numbers allocated to each value. Defining variables in an array makes it easier to find a specific value in a long list of values.

There are three distinct ways to create an array. They are regular, condensed and literal. They will all return the same result but look different in coding. Only the regular array requires the use of index numbers which follow the variable and are contained in brackets ( [ ] ). There are many more options available when creating arrays that will be covered later in the advanced chapters of this manual. Index numbers always start with the number zero “0”.

Regular Array:	<pre>var myFriends=new Array(); myFriends[0]="John"; myFriends[1]="Tom"; myFriends[2]="Steve";</pre>	Creates a list containing the values John, Tom and Steve that is selectable using the index numbers.
Condensed Array:	<pre>var myFriends=new Array("John","Tom","Steve");</pre>	Creates a list containing the same values as the regular array. Considered a form of shorthand.
Literal Array:	<pre>var myFriends=["John","Tom","Steve"];</pre>	Creates the same list as the previous examples and is even further shortened using brackets to delimitate the values.

# EXPLORING JAVASCRIPT DATA TYPES

## 5.7- Object:

In JavaScript an object is delineated by curly brackets (`{ }`). Inside the curly brackets are the object's properties. They are defined in name and value pairs, with the name and value separated by a colon.

**name:value**

You can list as many properties inside the curly brackets as you like, separating them with commas. Your declaration can span multiple lines and any extra spaces or line breaks are ignored. The following chapter covers much more about JavaScript objects.

<b>Example:</b>	<pre>var person = {   lastname:"Smith",   weight:190,   eyecolor:"Green" };</pre>
<b>Explanation:</b>	The example shows the object <b>person</b> with three properties of: <b>lastname</b> , <b>weight</b> and <b>eyecolor</b> .



# ACTIONS-

## EXPLORING JAVASCRIPT DATA TYPES

---

### CREATING (DECLARING) VARIABLES:

1. To create a text string variable, Type: `var x = "y";`  
Where "x" is the name of your variable and "y" is the value in quotation marks.
  2. To create a numerical variable, Type: `var x = y;`  
Where "x" is the name of your variable and "y" is the numerical value.
- 

### UNDEFINED VALUE:

1. To create a variable with an undefined value, type: `var x;`  
Where "x" is the name of the variable you are creating.
- 

### USING ONE STATEMENT FOR MULTIPLE VARIABLES:

1. Type: `var a="b", c="d", e=f;`  
**OR**  
`var a="b",`  
`c="d",`  
`e=f;`

Where "a", "c" and "e" are the names of your variables, and "b", "d" and "f" are the values.

---

### LOCAL AND GLOBAL VARIABLES:

1. To create a local variable, type: `function a() {var x="y"};`  
Where "a" is the name of your function, "x" is the name of your variable and "y" is the value of your variable.
  2. To create a global variable, type `var a="b";`  
Where "a" is the name of the variable and "b" is the value of your variable.
-

# EXERCISES-

## EXPLORING JAVASCRIPT DATA TYPES

---

### Purpose:

1. To add a variable to your JavaScript code.
- 

### Exercises:

1. Open the HTML document my-practice-page.html, created in a previous chapter.
2. At the end of the last line, before the </script> tag, press “Enter”.
3. Type: `var person=“John Smith”, weight=180, eyecolor=“Blue”;`
4. Press “Enter”.
5. In your text or HTML editor, Select “File| Save” from the Menu Bar or Ribbon, or use the keyboard shortcut CTRL+S to save your document

Sample- for evaluation purposes only!  
Visit us at [www.teachucomp.com](http://www.teachucomp.com)