Mastering Introductory JavaScript Made Easy™ v.1.0

TEACHUCOMP, INC.®

...it's all about you

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

Table of Contents

Table of Contents

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

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.

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:

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.

Start Tag:	<script>
End Tag:	</script>
Code Block Start "Tag":	{
Code Block End "Tag":	}

Example:	<script>
	function myFunction()
	{
	document.getElementById("line").innerHTML="Are you sure?";
	document.getElementById("test").innerHTML="Yes, I'm positive.";
	}
	</script>

Explanation:	Runs the JavaScript function "**myFunction()**" that will change, or manipulate, two separate HTML elements simultaneously.

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, \
Don't Stop");
```

Is correct.

But

```
document.write \
("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 \

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.

CHAPTER 3-
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.

Start "Tag":	**//**
End "Tag":	None
Example:	**//** Write to element with ID of "line". **document.getElementById("line").innerHTML="Are you sure?";**
Explanation:	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.

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

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.

Start "Tag":	**//**
End "Tag":	None
Example:	**var person="Thomas" //** Sets variable to value of Thomas.
Explanation:	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.

Single Line Comment Start "Tag":	**//**
Single Line Comment End "Tag":	None
Multi-Line Comment Start "Tag":	**/***
Multi-Line Comment End "Tag":	***/**

Example:	**// document.getElementById("line").innerHTML="Are you sure?";**
	
	/*
	document.getElementById("head1").innerHTML="Heading One";
	document.getElementById("para").innerHTML="First Paragraph";
	***/**

Explanation:	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

CHAPTER 4-
JavaScript Variables

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;

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";

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.

Local Variable:	**function foo ()**
	{
	var x = 9;
	}

Explanation:	The variable **y** with a value of **9** is considered to be “local” as it is contained within a specific function, in this instance “**foo**”. When the function has been executed the variable will be deleted and irrelevant to any future functions or calculations.
Global Variable:	**var x=7;**
Explanation:	The variable **x** with a value of **7** 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

CHAPTER 5-
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 exits 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.

Without Decimals:	**var x=7;**	Will display the number 7.
With Decimals:	**var y=9.00;**	Will display the number 9.
Scientific Notation:	**var a=7e5;**	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' ";.

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:	**var myFriends=new Array();
	myFriends[0]="John";
	myFriends[1]="Tom";
	myFriends[2]="Steve";**
		Creates a list containing the values John, Tom and Steve that is selectable using the index numbers.

Condensed Array:	**var myFriends=new Array("John","Tom","Steve");**	Creates a list containing the same values as the regular array. Considered a form of shorthand.
Literal Array:	**var myFriends=["John","Tom","Steve"];**	Creates the same list as the previous examples and is even further shortened using brackets to deliminate the values.

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.

Example:	**var person =
	{
	  lastname:"Smith",
	  weight:190,
	  eyecolor:"Green"
	};**

Explanation:	The example shows the object **person** with three properties of: **lastname**, **weight** and **eyecolor**.

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