**INTRODUCTION TO JAVASCRIPT**

This is a basic tutorial for learning the fundamentals of JavaScript (JS). The basic concepts you will acquire from taking this course will be directly transferable to JavaScript and computer programming in general.

**THE JAVASCRIPT INTERPRETER**

JavaScript is an interpreted language, meaning it is a language that requires additional software to read and compile the instructions into code that the computer will understand. This process will be performed one line at a time within the JavaScript document until all lines of instruction have been completed. Knowing how JavaScript is interpreted is essential to write effective code.

The JavaScript interpreter is a piece of software that comes preinstalled in most web browsers. The high-level code is executed by the browser as soon as the browser encounters JS code or a reference to a JS file.

A reference to JavaScript may be found in a different location in the HTML document depending on the intention of the developer: at the beginning within the head element or somewhere within the body element. When the browser encounters a JavaScript reference, it will temporarily cease loading anything further until all of the JavaScript instructions have been completed. Only then will the browser continue loading the rest of the HTML page. It is important to reference JavaScript in the correct location of your web document to avoid producing unintended behavior.

**STEP: 9**

**Introduction to JavaScript (cont.)**

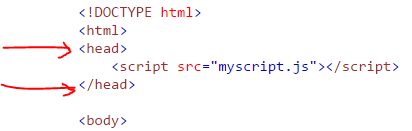
**HOW IS JAVASCRIPT USED?**

JavaScript is commonly thought of as a "supporting" language that merely adds the interactivity component to HTML and CSS documents. JavaScript can easily provide web-enhancements such as menu roll-over effects, photo-galleries, animation, productivity applications, and even games. Though JavaScript is certainly capable of providing interactivity between the user and the web page, JavaScript's true potential goes way beyond front-end web development.

**STEP: 10**

**HOW TO REFERENCE JAVASCRIPT**

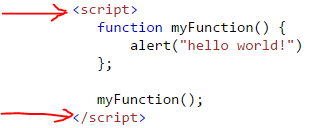
In order for JavaScript to be recognized by the browser, it must be properly introduced. There are a few locations where this reference can be performed. JavaScript may be referenced in the head element.



JavaScript may also be referenced from within the body element in a few ways. There could be a reference made to an external JavaScript page.

https://thetechacademy.azureedge.net/javascript/js_scripttag_remote.png

Between the script tags, inline JavaScript code could be written directly into the .html document.



What is important to note is that your JavaScript will, in most situations, interrupt the .html document flow as soon as it encounters a reference to your JavaScript file or code. The browser will attempt to execute or load the JS code first before continuing on to load the rest of the .html document. Therefore, it is highly recommended to place your code near the bottom of the document just before the closing body tag.

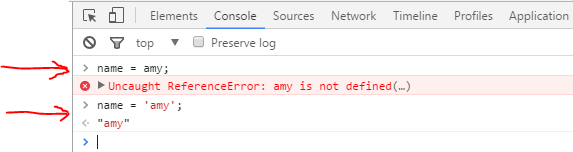
**STEP: 14**

**JavaScript Basics**

**DEBUG YOUR CODE AND SELECTING AN IDE**

A great place to test snippets of your JavaScript code is via the browser's console. If you are unsure exactly if your code will produce an error or not, try placing it in a browser's console and see what gets returned.

In the following example, my code had produced an error because I had forgotten to use quotation marks around my string value, so the interpreter thought that I was referencing a variable that had not yet been declared. After plugging my code into the console, I was quickly able to notice this error and correct it.



**STEP: 15**

**JavaScript Basics (cont.)**

There are other great resources to assist you while you code your JavaScript. The Text Editor provides the bare-minimum that would be needed to write JavaScript. Some Text Editors are more advanced offering splitscreen views, syntax highlighting, line numbers, and other helpful features.

Another important tool for a software developer is an Integrated Development Environment (IDE). An IDE is a powerful tool that can help keep track of project assets. It also offers preloaded templates, code snippets, syntax highlighting, debugging tools, and built-in functionality such as Intellesense. Intellesense is a feature which attempts to offer code corrections, suggestions, and even code completion depending on the event it has encountered. Learning to use and master an IDE is highly recommended.

In addition to an IDE, there is the Real Eval Print Loop (REPL). This is a fancy way of saying that it is a piece of software shell that interprets code in real-time allowing a coder to instantly see the results of their code. The Browser console is one such example of a REPL, but there are some notable text editors that can be enhanced with additional scripts for a full-page REPL instead of just a single line-by-line REPL like the browser's console. What is very important to note is that an REPL is not an IDE and will not offer convenient features like Intellesense, code completion, and syntax highlighting.

Text Editors, IDE, and REPL may all be installed on the client computer or accessed via cloud-based services. Among the most notable freeware Text Editors, IDE, and REPL software are:

[**Microsoft's Visual Studio**](https://visualstudio.microsoft.com/)

[**Sublime Text**](https://www.sublimetext.com/)

[**Brackets**](http://brackets.io/)

[**Node.REPL tools**](https://nodejs.org/en/)

[**JSFiddle**](https://jsfiddle.net/)

[**Repl.It**](https://repl.it/)

[**Codepen**](https://codepen.io/)

**WHAT IS THE DOM?**

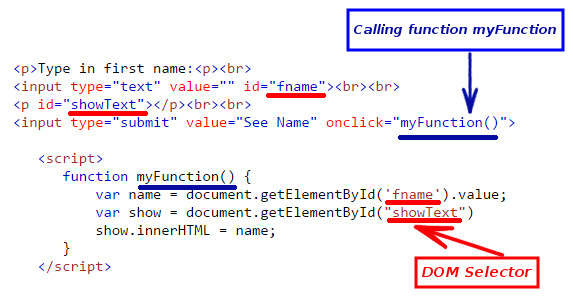
Before you can understand what the DOM is, you will first need to understand Application Programming Interface (API). API is a package of defined processes, rules, and devices that programmers can use to get their software to work together. Developers can define the necessary routines, protocols, syntax into a neat package called API. The developer can then make their API available to other developers.

The Document Object Model (DOM) is an application programming interface (API) for the HyperText Markup Language (HTML). JavaScript harnesses the DOM to access and manipulate virtually anything within the HTML document. By using the HTML API, JavaScript serves as the master language for front-end design, as JavaScript has the ability to programmatically build HTML elements during runtime and then affect their associated CSS values or even redefine those values altogether. JavaScript can be developed to detect browser events such as page scrolling and to an extent, control the browser.

Employing the DOM API, JavaScript is able to target specific HTML elements, much in the similar manner that CSS would. This process is termed, DOM selecting. Using DOM object selectors in JavaScript is crucial to DOM manipulations. Selectors are the keys to the world of HTML and CSS. Once a DOM element has been selected, a developer can affect all kinds of interesting and useful front end events.

**STEP: 20**

THE FOLLOWING IS AN EXAMPLE OF JAVASCRIPT CODE "DOM SELECTING" HTML ELEMENTS:



There are many overlapping similarities that programming languages share. Most programming languages will work with data in some fashion or another and thus needs to be appropriately classified. Each classification of data will have its own characteristics and behaviors and unique methods for storing that data. These classifications are termed Data types, and understanding data types enables programmers to design their applications without errors or data loss.

**STEP: 21**

**JavaScript Basics (cont.)**

JavaScript uses a few primitive data types, String, Numeric value, Boolean, NULL, and Undefined.

**String values:**are collections of characters placed between two quotation marks.

**Numeric values:**are numbers that are not placed in between quotation marks.

**Boolean values:**are either a True or False state.

**Null values:**is a special value type, indicating no value at all.

**Undefined values:**are assigned by the interpreter automatically due to an illegal operation where neither a value nor a data type has been specified. As a result, the interpreter is not certain how it should proceed.

**STEP: 22**

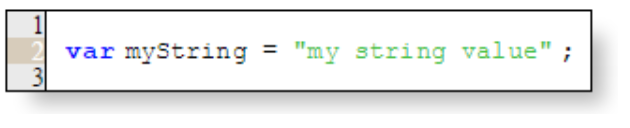
**VARIABLES:**

**JavaScript Basics**

Variables maintain a persistent reference to the data it has been assigned. Abstractly, the variable can be thought of as a container that may hold a specific data type. What is actually occurring on the back end is that a portion and position of a computer's memory is being allocated to store the value assigned to that variable. The variable merely serves as a marker to the corresponding data nestled with memory.

For all intents and purposes, it is much easier to simply visualize a variable as a box that will hold a particular value. And at anytime, that value can be viewed, accessed, overwritten, or the variable along with the value can be removed from memory altogether.

Defining variables within JavaScript is fairly painless. You would first need to declare to JavaScript your intentions to create a variable with the var keyword, followed by the variable name.



Here is an example of a variable being declared and assigned a string datatype.

There are many other programming languages that would have further requirements such as a declared data type, but JavaScript is capable of determining the data type without specifically being told. It can tell what data type it is as soon as you've completed assigning the variable.

**STEP: 23**

**JavaScript and Arrays**

Another useful data object of JavaScript is the array. An array serves as a container that stores similar data objects. For instance, this could be a list of images, hyperlinks, strings, hexadecimal numbers, mp3 files, etc. The array object is a very powerful feature available in JavaScript.

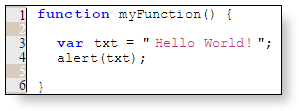
https://thetechacademy.azureedge.net/javascript/declare_array.png

Here is an example of an array being declared and assigned with several string values.

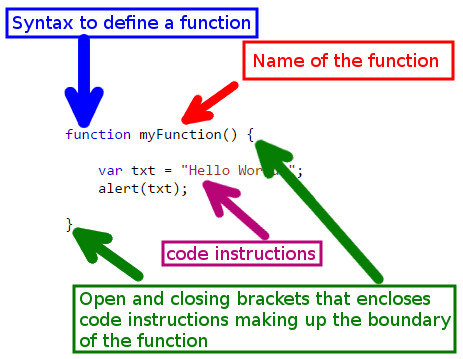
**STEP: 24**

**THE JAVASCRIPT FUNCTION STATEMENT**

An essential aspect of Object-Oriented Programming is the Function. Functions are specialized blocks of code that perform specific tasks and may be called upon repeatedly within the scope of the application. Functions are vital to JavaScript, so it is important to understand how they are declared, constructed, and invoked.



This is an example of a basic function statement.



This is the anatomy of that very same function statement.