What is Javascript

Lesson Time: 10 Minutes

Our first taste of programming.

Unlike CSS and HTML, which are markup languages, Javascript is a programming language. Javascript can do things such as add 2 + 2, tell if a statement is true or false, or repeat a set of commands over and over again until a task is complete. The goal of all programming languages is to process data--a language accepts input, changes it, and creates output.

Javascript is one of the most widely used programming languages in the world, and dominates the frontend of web development. Every web browser has a built-in “engine” for processing Javascript code.

Javascript is becoming more popular on the backend, due to new technologies like Node.JS. However, Javascript is still primarily used on the frontend, and much less on the backend. In our full stack, we will use Javascript in the browser and C# / ASP.NET on the backend.

Javascript lets us do these **three** things in the web browser:

1. Change the style and content of the page & respond to user actions.
2. Send data to and from the web server in the background, ***without*** reloading the page.
3. Implement features in our web app and makes the web browser responsible for those features *instead* of the web server. This is called **client-side programming.**

Key Terms	
Lesson Files	
Additional Resources	
Further Learning	**You might not need Javascript** A guide to using pure CSS to pull off web effects http://youmightnotneedjs.com/

Javascript is not Java

Lesson Time: 10 Minutes

Let’s call it JS.

One point of confusion is that there are two programming languages out there, Javascript and Java. They are not the same!

From this point forward, we will refer to Javascript as just JS. JS is used in web browsers as a frontend programming language. Java is a general purpose language that can be used to create desktop applications. Java can also be used as a backend language for the web server, however, there are many other languages that compete with Java on the web server. Javascript, PHP, & C# all are languages that can be used in backend web servers. In this course, we use C# in our tech stack.

If you are wondering why we ended up with this confusion, blame the marketing team at Sun Microsystems. JS was originally called LiveScript. When JS was being developed, Java was a new and hot language everyone wanted to learn, and when Sun partnered with Netscape, they rebranded it as JavaScript to ride on Java’s popularity.

Key Terms	
Lesson Files	
Additional Resources	https://www.seguetech.com/java-vs-javascript-difference/
Further Learning	

<script> tag

Lesson Time: 10 Minutes

Where Javascript is placed on a web page.

The HTML <script> tag holds JS code that will run in the browser. JS can be written directly in the html file inside of the script tag, or saved in a seperate .js file. When saving the script in a seperate .js file, the script tag will point to the location of the script file.

Internal JS Script embedded in HTML

```
<head>
  <script>
  document.getElementById("demo").innerHTML = "Hello World!";
  </script>
</head>
```

External JS Script linked to HTML

```
<head>
  <script src="myScript.js"></script>
</head>
```

The script tag may be used in either the **head** *or* the **body** of the html. When placing a script in the body, it is best practice to place it ***at the bottom of the body*** to allow all other HTML elements to load before running javascript. JS code will execute one line at a time, from the top to bottom until the script reaches the end.

Key Terms	<script>
Lesson Files	
Additional Resources	https://www.w3schools.com/js/js_whereto.asp

Further Learning	

JS Variables, Constants, and DataTypes

Lesson Time: 60 Minutes

Programming Languages are all about processing data.

When you create a **variable** in javascript, you are creating a placeholder in computer memory that stores data. You can assign a value to the variable, and then change that value at any time while the script is running.

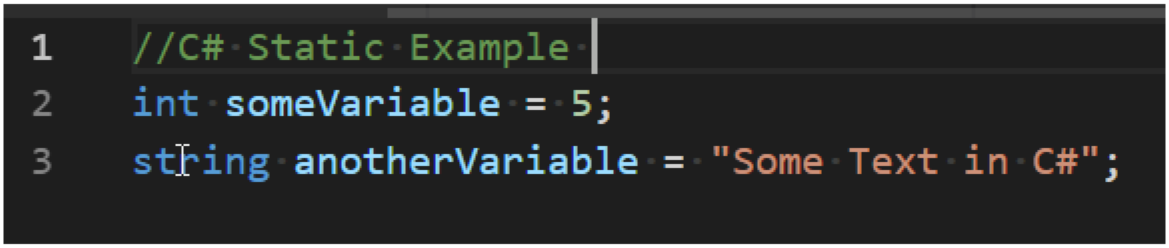
Javascript variables are created by using either the keyword **`var`** or the keyword **`let`**. Variables declared with **var** will behave slightly differently that variables declared with **let**. We will cover this difference when we are reviewing SCOPE. Keywords in JS are always lowercase. JS is case-sensitive.

Dynamically Typed

Variables in javascript are **dynamically typed**. This means is that you do not have to tell Javascript what kind of you are working with--JS just knows. For example

```
var someVariable = 5
var anotherVariable = "This is some text."
```

Javascript understands that *someVariable is a number* and that *anotherVariable is a string of text*. You do not have to specify this to Javascript. This is what it means when a language is dynamically typed. The opposite of a dynamically typed language is a **static typed** language. In a static typed language, you must include what type of data the variable is. An example is this line of C#.



```
//C# Static Example
int someVariable = 5;
string anotherVariable = "Some Text in C#";
```

Case Sensitive

Javascript is case-sensitive. Let’s look at errors that can happen due to case problems.

```
var someText = "this is my variable"
console.log(SOMETEXT)
//this errors because the SOMETEXT is not the same as someText

VAR someText = "this is my variable"
console.log(someText)
//this errors because VAR is not the same as var

var someText = "this is my variable"
console.log(someText)
//this runs because var is all lowercase and the variable name someText is
correct
```

Mutable / Immutable

The data stored in variables can be changed at any time. For example, the following code is valid:

```
var data = 7
data = 10
console.log(data)
```

When data can be changed, we say the data is **mutable**.

Constants are created with the keyword `const` . Constants are different from variables in that once their values are assigned, *the values can not be changed*. You would want to create a constant when you need some data to never change in the script and you want to protect it from accidentally being changed. When data cannot be changed, it is called **immutable** .

Data Types

Every variable and constant has a **data type.** The basic data types in JS are

- **number (Any number, including 0, -1, and 5.25)**
    - **In JS, even a date like 12/01/1980 is a number**
- **string (text)**
- **Boolean (true or false)**
- **Undefined**

Undefined is slightly tricky. Undefined variables means the variable has been created but no value has been assigned. Since no value is assigned, JS can not tell if it's a number, string, or true/false, so it is simply "undefined".

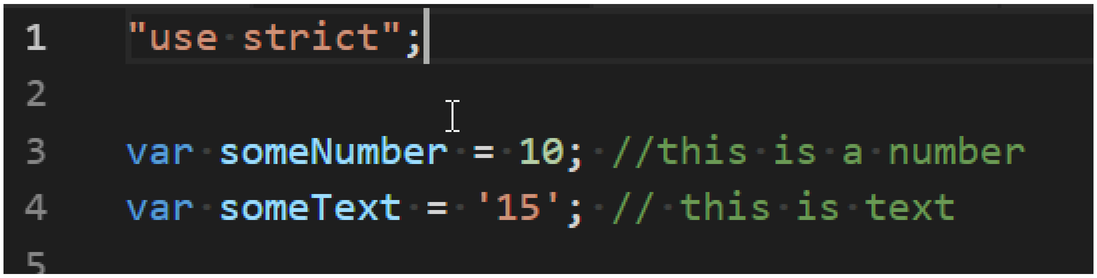
In JS, undefined is slightly different from null. A **null** value is an empty value or a "nothing" value. In JS , a variable can be assigned a null value. However, by assigning it null, it is no longer undefined--the value has been assigned, even if it's value is "nothing".

Use Strict

The first line of your JS script should always begin with "use strict". This turns off certain old behaviors of JS and turns on modern JS. "use strict" will help prevent bugs in your code. The only time you would not include it is when working with very old JS scripts. However, you must remember to start the script with it.

Semicolon or No Semicolon?

JS requires a semicolon at the end of a line statement. However, JS is so smart that when you don't include it, it assumes you meant to, and nothing bad happens. 99% of the time, if you don't end your line of code with a semicolon everything will be fine. It's best practice to include it, and developing the habit prepares you for other languages like C#. Just know, if you forget it, JS is very forgiving and shouldn't error out.



```
"use strict";

var someNumber = 10; //this is a number
var someText = '15'; // this is text
```

Key Terms	Var, let, const, mutable, immutable, static typed, dynamic typed, undefined, null, string, boolean, data type
Lesson Files	
Additional Resources	
Further Learning	