

Web Scripting 1

Day 01

Agenda

- Course Information & Administration
- Project 01
- Introduction to JavaScript
 - What JavaScript can and can't do
 - Syntax
 - Troubleshooting
 - Variables
 - Window methods
 - Events
- Assignment 01
 - Using Live Share to collaborate on coding projects

About the course

- **Objectives:**

- Provide an introductory look at JavaScript
- Introduce students to common computer scripting operations and terms such as variables, loops, arrays, conditionals and objects
- Give students the knowledge to be able to add common interactive behaviors and functionality to their web sites such as slide shows, lightboxes, modals, animations and form validation
- Call an API
- Validate a form

- **Format:**

- Some theory, mostly practical, hands on
- Morning lecture, with afternoon lab time
- 6 assignments
- 1 project

Daily Schedule

- Each class is 6 hours
- Most classes will be divided up as follows
 - 9am – 12:30pm
 - 0.5 hours of review from previous week
 - 2.75 hours of lecture
 - New material
 - In-class exercises
 - 15 minute break (approximately 10:45am – 11am)
 - 1pm – 4pm
 - Instructor supervised open lab

What you will learn

- Write basic JavaScript for running in a modern web browser
- Introduction to common JavaScript scripting operations and terms such as variables, functions, loops, arrays, conditionals and objects
- Creating and adding common interactive behaviors commonly found on modern web sites such as slide shows, lightboxes, modals, animations and form validation
- Call an API
- Validate a form

The Many Different Levels of Abilities in FWD

- Be aware that the FWD program attracts students from a wide variety of backgrounds
 - Many of your fellow students will have no background in computer programming concepts such as variables, loops and arrays
 - Some of your fellow students may have years of computer programming experience in other languages
- If you are new to programming
 - Don't be discouraged. Grasping computer programming basics can take time
 - The first few days of the course may be a bit of a struggle, but almost everyone will be successful in learning JavaScript. I am 100% confident that you will get there!
 - Never be afraid to ask questions, either in class or in confidence to me via Slack, email, or in person

JavaScript is Your First Language

- If you are completely new to programming
 - Don't be discouraged. Grasping computer programming basics can take time
 - The first few days of the course may be a bit of a struggle, but trust me, the concepts will start to make sense after a bit of practice
 - Don't be alarmed if you see other students seemingly mastering the assignments and exercises with little or no effort. They more than likely have experience with other computer programming languages
 - Don't worry if you hear questions from other students that talk about computer programming terms that you may not be familiar with. Again, these students probably have experience with other languages and are just looking for comparisons to other languages. We will cover many of these common programming terms as the course progresses

JavaScript is my 2nd, 3rd...etc...Language

- Be courteous and respectful of your fellow students who may not know many introductory computer programming and scripting fundamentals
- This course is an introductory course, so expect some review of basic computer scripting concepts

Additional Learning Materials

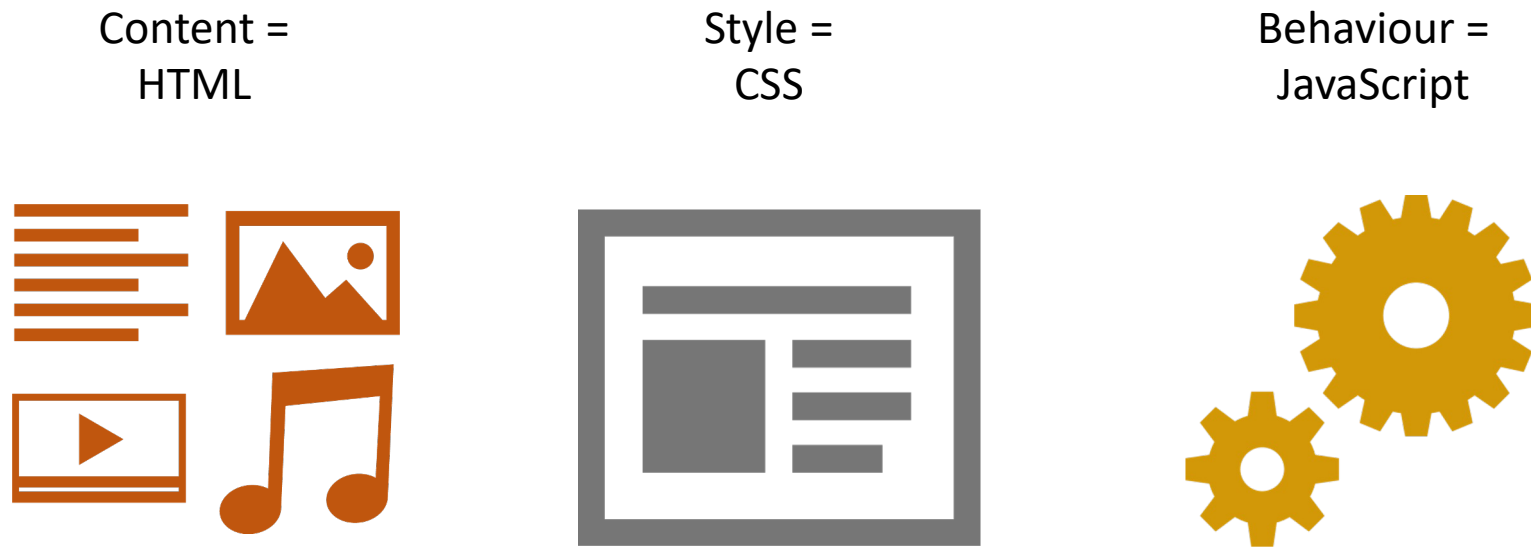
- There is no textbook for this course, but I will provide links to additional online resources for optional reading
- Many additional online resources for learning about web development
 - Web sites
 - <https://developer.mozilla.org>
 - <https://stackoverflow.com/>
 - <https://css-tricks.com/>
 - <https://www.w3schools.com/>
 - YouTube
 - Great place for free video learning on a variety of web development topics
 - It's free, so quality can vary
 - Here are some good channels for learning:
 - [The Net Ninja](#)
 - [Traversy Media](#)
 - [LearnWebCode](#)
 - LinkedIn Learning
 - As a BCIT student you have access to the entire Lynda Library for free
 - Courses are generally of a high quality
 - Search for [your language of choice] + Essential Training (eg: JavaScript Essential Training) for introductory courses
 - Here is a link to the JavaScript Essential Training Course on Lynda: [JavaScript Essential Training](#)

JavaScript

An Introduction

JavaScript that Runs in the Browser

- JavaScript is the behaviour language of the web
 - Handles behaviours that occurs on a web site



What is JavaScript

- JavaScript is the only scripting language that is native to the web browser
 - No browser plugins or extensions are required to run JavaScript in the browser
- Shares structured programming syntax from C¹
 - "if", "while loops", "switch" etc. will have similar syntaxes
- Dynamically typed
 - A data type for a variable is assigned by the value of that variable dynamically by JavaScript
- Almost entirely object based
 - Most elements in JavaScript are objects

1. <https://en.wikipedia.org/wiki/JavaScript>

What JavaScript is NOT

- JavaScript is NOT Java
- JavaScript is NOT a light version of Java
- JavaScript has no relation to Java, they are completely separate languages



What JavaScript Can Do (within the Browser)

- Dynamic animations that run based on user interaction
- Inject new content into a document dynamically
- Update, change, delete and move existing content on a web page
- Validate forms
- Trigger code based on browser events such as scrolling, clicking, resizing, mouse movements and other events
- Browser based video games

What JavaScript Can't Do (within the Browser)

- The following points are only in regards to JavaScript that runs in a browser. Other flavors of JavaScript such as Node.js may not have these restrictions
- JavaScript can not access a user's file system
 - Reading, writing, deleting and changing a user's files stored in the native operating system is not possible
- Change or effect code located on other external websites
 - You can not write JavaScript that will affect another web site

JavaScript Can Be Disabled by the User

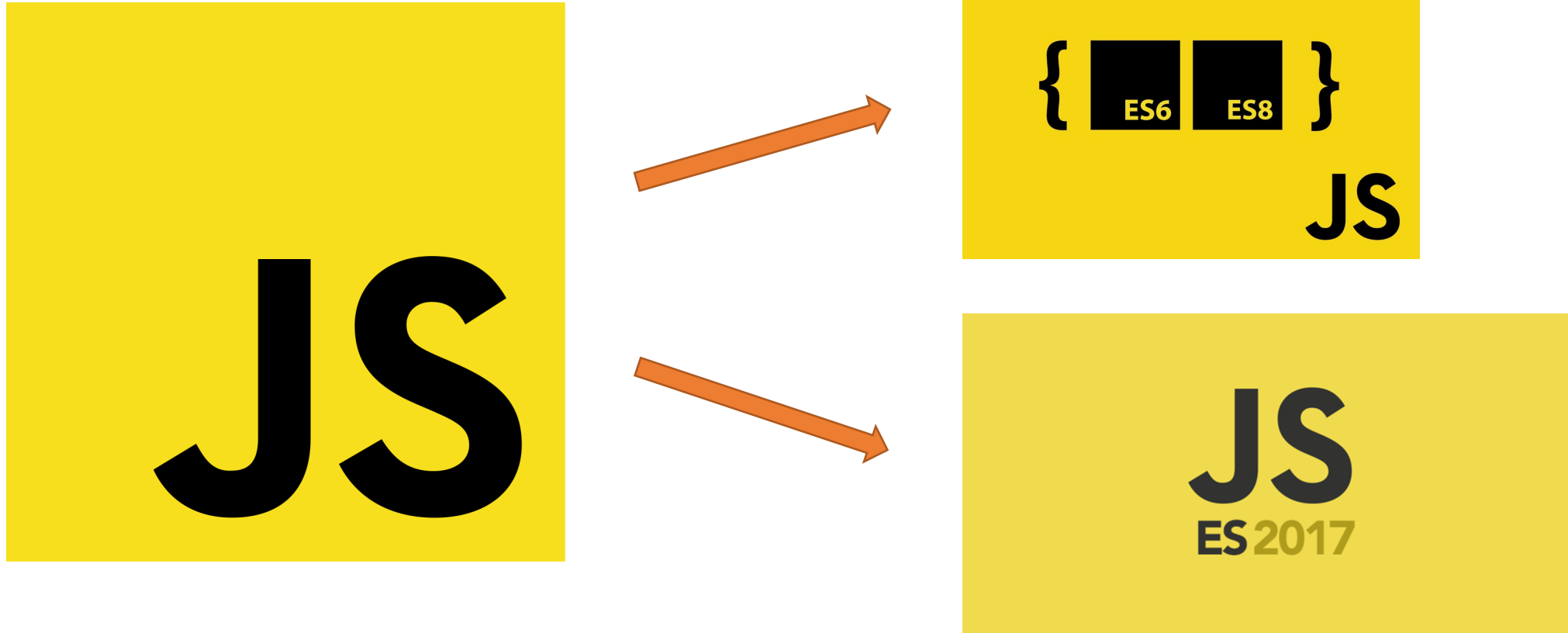
- Be aware that a user can disable JavaScript in the browser by changing a simple browser setting
 - If they do, none of your JavaScript code will run
 - You can not force your users to run JavaScript
- The vast majority of people leave JavaScript enabled, so this is only a minor concern
- If your web app or web site requires JavaScript to run, you can display a message inside a "<noscript></noscript>" tag and ask your users to enable JavaScript

ECMA Script vs JavaScript

- Ecma
 - Used to be based on the following term:
 - European Computer Manufacturing Association¹
- ECMA Script is a scripting language specification that was created by Ecma International to standardize JavaScript
- ECMA Script is just a set of standards that the JavaScript language is based on

1. https://en.wikipedia.org/wiki/Ecma_International

The Versions of JavaScript



1. ES6 & ES8 Image: <https://hackernoon.com/7-different-ways-to-use-es-modules-today-fc552254ebf4>

2. JS ES2017 Image: <https://medium.com/komenco/what-to-expect-from-javascript-es2017-the-async-edition-618e28819711>

The Versions of JavaScript

- EcmaScript 2009 – aka: ES5
 - Older version of JavaScript
 - Many consider this the baseline version of the language that all modern browsers support
 - Some older online tutorials will use this older pre-2015 syntax
- ES2015, ES2016, ES2017, ES2018 – aka: ES6, ES7, ES8, ES9
 - Major revision of the language (ES2015)
 - Added block scoped variables (const and let) to the function scoped "var" based variables
 - New ways for writing functions
 - Pseudo "Classes" added to the syntax (not true classes, really just syntax sugar)
 - ES5 syntax will work in JavaScript code written in a modern version of JavaScript
 - The foundational aspects of the language remained unchanged from ES5
 - Most modern web browser support most (but not all) the new features of the language introduced with ES2015 (ES6)

Future Versions of JavaScript

- Ecma has decided to adopt a yearly release cycle for JavaScript
- Expect new features of JavaScript to be introduced on an annual basis
- Research what new features of JavaScript your target browsers support
 - It takes time from when new features of JavaScript are released to when browsers implement them

This course will use ES2015 (ES6) Syntax

- We will use the ES2015 (and newer) syntax of the language for this course
 - The code we will use in-class may differ slightly from older online tutorials you may find on the web
 - Key Differences
 - In this course we will use "const" and "let" to declare variables, whereas many online tutorials may still use the older (but still supported) "var" keyword to declare variables
 - We may use arrow functions for declaring some anonymous functions
 - You will see syntax like this for declaring anonymous functions (ES2015 arrow syntax):
 - `_ => {}` or `() => {}`
 - Which is similar (but not exactly the same) as writing an anonymous function like this (ES 2009 syntax):
 - `function(){}`
 - **Note:** The above syntax is still supported in modern versions of JavaScript and may even be required in some instances

Online JavaScript Learning Resources

- MDN (Mozilla Developer Network) - JavaScript
 - The user manual for JavaScript
 - <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- W3Schools
 - Great for beginners
 - <https://www.w3schools.com/js/default.asp>
- LinkedIn Learning
 - As a BCIT student you have free access to LinkedIn Learning
 - Courses are professionally produced and of a high quality
 - JavaScript Essential Training by Morten Rand-Hendriksen is a good intro to JavaScript course
 - <https://www.lynda.com/JavaScript-tutorials/JavaScript-Essential-Training/574716-2.html>

Online JavaScript Learning Resources (continued)

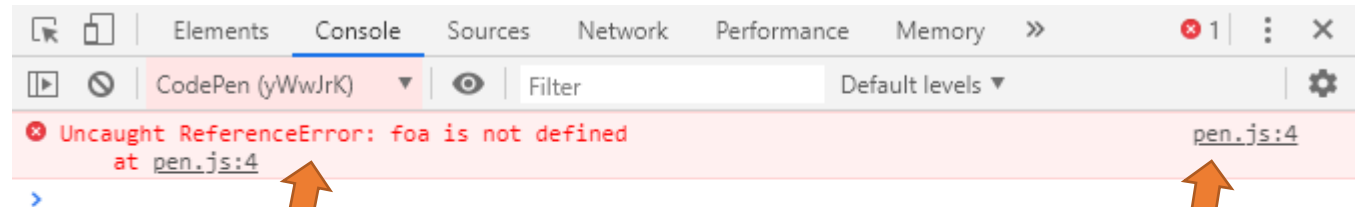
- YouTube
 - Lots of free "How to" videos on YouTube about JavaScript
 - Quality can vary
 - Some good channels to checkout
 - Traversy Media
 - <https://www.youtube.com/channel/UC29ju8bIPH5as8OGnQzwJyA>
 - The Net Ninja
 - <https://www.youtube.com/channel/UCW5YeuERMmInqo4oq8vwUpg>
 - Check out his JavaScript for Beginners Playlist
 - <https://bit.ly/2qGJBmF>
 - LearnWebCode
 - <https://www.youtube.com/user/LearnWebCode>

Browser Developer Tools for JavaScript

- JavaScript will fail silently, meaning a web page that also has JavaScript will still load, even if a script is broken or has a syntax error
- The Browser Developer Tools Console tab is the place to go to look for JavaScript errors
- Right clicking on a web page and selecting "Inspect" or "Inspect Element" will usually bring up the browsers developer tools
 - Safari has to have its developer tools enabled in the settings
 - For Microsoft Edge browser you can press the F12 key, if the "Inspect Element" is not present in the right-click menu
- Once you have the browser's developer tools window open, look for the "Console" tab and click on that tab to show the JavaScript console window

The Browser JavaScript Console

- The console will report script errors that the browser found while trying to execute your JavaScript code
- The reported errors will report the error type, the error and the file and code line number where the error occurred

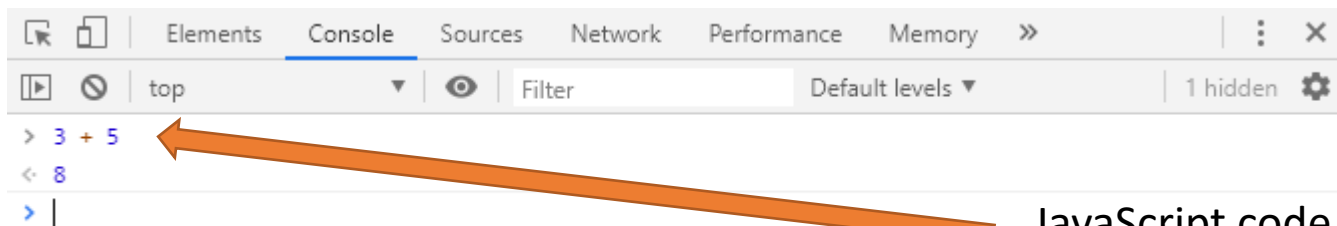


The error type and description

The location of the error

Running Code in the Console

- You can run a single line or multiple lines of JavaScript code from within the console by entering code directly into the console
 - This code is not saved and will be deleted once you close the developer tools session
 - To run a single line of code press the "Enter" key after you write your code
 - To run multiple lines of code (Chrome Browser) press "Shift+Enter" (Windows) or "Option-Enter" to add an additional line



JavaScript code
entered into and ran
in the console

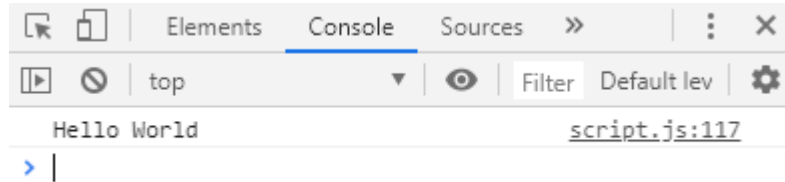
Logging Text or Data to the Console

- You write messages to yourself (or other developers) that display in the browser's console window
- Use the `console.log('Your message...')` to output messages to the console
- You can also output a variables value to the console by simply writing the variable's label as an argument to the `console.log()` method
- You can also combine a message with a variable value in a `console.log()` message

console.log() Examples

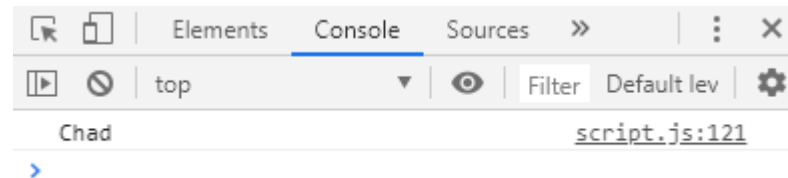
Output a message

```
// Output a text message to  
// the console  
console.log('Hello World');
```



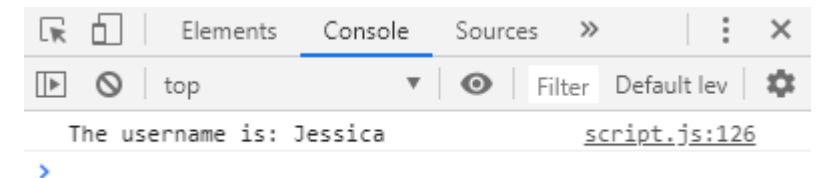
Output a variable

```
// Output a variable...  
const username = 'Chad';  
console.log(username);
```



Output a message with a variable (this is one way of doing this, there are other ways to do this as well)

```
// Output a message  
// with a variable  
const username = 'Jessica';  
console.log('The username is: ' + username);
```



Adding JavaScript to Your Web Page

- Three ways to include JavaScript on a web page
 - Inline
 - JavaScript is written inside an element as an event handler
 - I personally do not recommend this approach
 - Makes for messy HTML
 - Mixes HTML code with JavaScript code
 - Usually best to separate different code into different files
 - Embedded
 - JavaScript code is embedded into the HTML page directly by writing JavaScript between "<script></script>" tags
 - Good solution for simple one page site's and simple short scripts that only need to run on a single page
 - External
 - Recommended method for most scripts
 - JavaScript is written in separate ".js" files and attached to the HTML by adding a "src" attribute to a "script" element

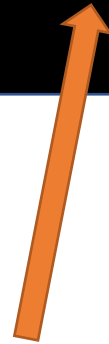
Inline JavaScript

- You can write event handlers as an attribute of an HTML element
- I personally do not recommend this method for a few reasons
 - Mixes concerns
 - HTML/CSS is the view and JS is the logic, it is often consider best practice to separate logic code from view code
 - Can be more difficult to troubleshoot JavaScript errors that are inline with the HTML
 - Impossible to reuse JavaScript code on multiple pages
 - If you use a version control system or source code management system, having different types of code in the same file can make it more confusing to track changes
 - Can make the JavaScript difficult to read

Inline JavaScript Syntax

- JavaScript written inside an HTML tag (not recommended)

```
<button onclick="alert('Hello World')">Click Me</button>
```



This inline JavaScript code adds a "click" event handler that will show an alert pop-up box with the text of "Hello World" when the button is clicked

Embedded JavaScript

- JavaScript code written in between "<script></script>" directly inside the HTML file
- Advantages:
 - Good solution for small single page web sites
 - Good solution for small scripts that only need to run on a single page
 - The "<script></script>" tags can go anywhere in the HTML file, but are most often placed in either the "head" section of an HTML file or just before the closing "</body>" element
- Disadvantages:
 - Different languages mixed into the same file
 - Prevents browser caching of scripts
 - Impossible to reuse scripts across multiple pages

Embedded JavaScript Syntax

```
<button id="btn-say-hello">Click Me</button>

<script>

document.getElementById('btn-say-hello')
    .addEventListener('click', _ => alert('Hello World') );

</script>
</body>
</html>
```



This embedded JavaScript code adds a "click" event handler to the button with an "id" of "btn-say-hello". The "click" event handler will display an alert pop-up box with the text of "Hello World"

Why Place Scripts Before the Closing Body Tag

- Browsers read HTML documents from top to bottom. This includes scripts
- If a script is placed in the head section before any content HTML in the body and the script looks for HTML elements in the body element an error will occur because the body has not been loaded when the script is run
- Scripts can block the running of content HTML until the entire script has been parsed which could make your page appear less performant
- The above issues can be mitigated by using "async" and/or the "defer" attribute on the script tag
- An alternative solution is to place your script tags immediately before the closing body tag

External JavaScript Syntax

- JavaScript is written in external text files saved with the ".js" extension
- JavaScript files are attached to the HTML via a "src" attribute on a "<script></script>" element

```
<script src="scripts/script.js"></script>  
</body>  
</html>
```




The external JavaScript file "script.js", which is located in a "scripts" folder is attached to an HTML document via a "src" attribute placed on a "<script></script>" element


JavaScript Syntax

- JavaScript is an object based language that uses dot "." syntax to call methods (things an object can do) and properties (things that describe an object)


```
document.getElementById('p-01').innerHTML = 'Hello World';
```




This is the "document" object. This object is created for you by the browser and represents the HTML document



We use the "." followed by the method or property name to access a method or property. Here we are calling the "getElementById" method on the document object. Methods have "()" after their name



Objects can be chained. In this case the "getElementById" method selects an HTML element which is itself a new object that has an "innerHTML" property



Here we are changing the value of the "innerHTML" property on the "p-01" HTML element to "Hello World"

JavaScript Syntax

- JavaScript is case sensitive

~~document.getElementById('p-01');~~

BAD!!! This will not work. Properties, methods, objects, variables and keywords must match their case exactly

document.getElementById('p-01');

GOOD!!! This example calls the "getElementById" with the correct casing

What is Camel Case?

- Most built-in methods and properties in JavaScript are written in camel case
- To write a variable, method, property or function in camel case you write the first word in all lowercase and then the first letter in subsequent words are uppercased

```
const billTotalAfterTax = 23.57;
```



First word is all lowercased

The first letter of subsequent words are uppercased

Camel Case vs Snake Case

- For your own custom variables, functions, methods and objects you do not have to write them in camel case, and can instead separate words with an "_" (underscore) character, sometimes called "Snake case"
- Most JavaScript developers will write their own custom variables, functions, methods and objects in camel case

Snake Case

```
const bill_total_after_tax = 23.57;
```

- Less common with JavaScript developers

Camel Case


```
const billTotalAfterTax = 23.57;
```

- More common with JavaScript developers

The Semi-Colon

- Unlike many other languages, the ";" character at the end of a line of JavaScript code, called a "statement" is usually optional
- The choice to either add a semi-colon at the end of a JavaScript statement or not use a semi-colon is an endless debate
- Personal Opinion
 - No right or wrong answer here
 - I personally always use a semi-colon at the end of all my JavaScript statements
 - Feel free to make your own decision on this one

```
document.getElementById('p-01').innerHTML = 'Hello World';
```



The semi-colon at the end of a JavaScript statement is usually optional

JavaScript Code Comments

- Comments are notes in your code that are ignored by the JavaScript parser
- Comments can be used for:
 - Explaining code blocks to other developers or to your "future" self when looking at your code several months later
 - Temporarily turning off blocks of code by commenting it out for troubleshooting or other development needs
 - Script version numbers
 - Providing attribution information for code sourced from other places
 - Copyright and license information

JavaScript Code Comments are Public

- JavaScript code is not compiled, it is delivered as is (in plain text) directly to the browser
- Anyone with a browser can view all your JavaScript code
 - Even if you minify your code, it is still a one click operation to "prettify" your minified code and then view it directly in the browser
- Since your JavaScript code is easily viewable by anyone avoid the following in your comments:
 - Passwords and login information
 - User passwords and logins
 - Private information that you do not want public
 - Probably best to avoid offensive content in your comments

JavaScript Comment Syntax

- With JavaScript you can write multiline or single line comments
- Multi-line comments are written with `"/ * ... your comments... */`
- Single-line comment are written with a `"// ...single line comment"` before each comment line

Multi-line Comment

```
/*  
  
- This is a multiline comment  
- All text between the opening  
  comment marker and the closing  
  comment marker will be ignored  
  
*/
```

Single-line Comment

```
// This is a single-line comment
```

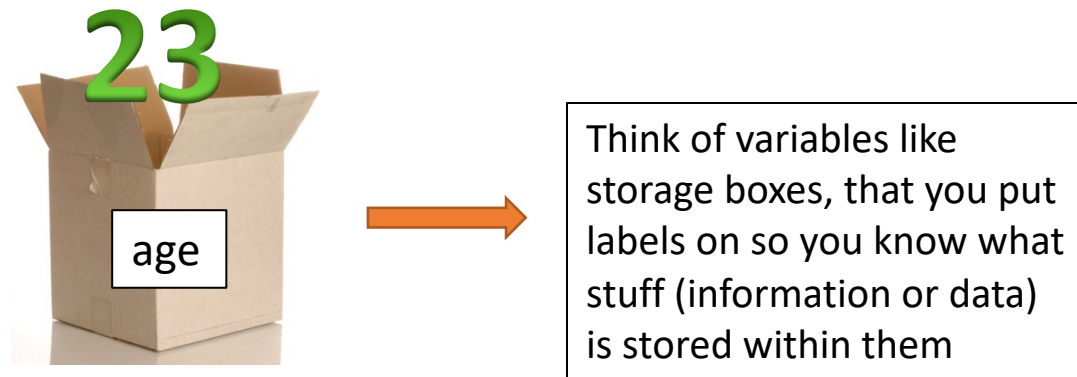
Visual Studio Code Comment Short Cut

- Visual Studio Code (and most other code text editors) allow for quickly writing comments by pressing "Ctrl+/" (Windows) or "Cmd+/" (Mac)
 - This works in HTML and CSS as well
- Another trick for quickly commenting out blocks of code is to select the block of code you wish to comment out and then press "Ctrl+/" (Windows) or "Cmd+/" (Mac)

JavaScript Variables

What is a Variable?

- A variable in computer programming is used to store information (data) to be referenced (retrieved) and manipulated in a computer program¹
- Developers use descriptive variable names as labels on data for easier retrieval of the data and to aid in understanding what kind of data is being stored in the variable¹



1. <https://launchschool.com/books/ruby/read/variables>

Variable Scope

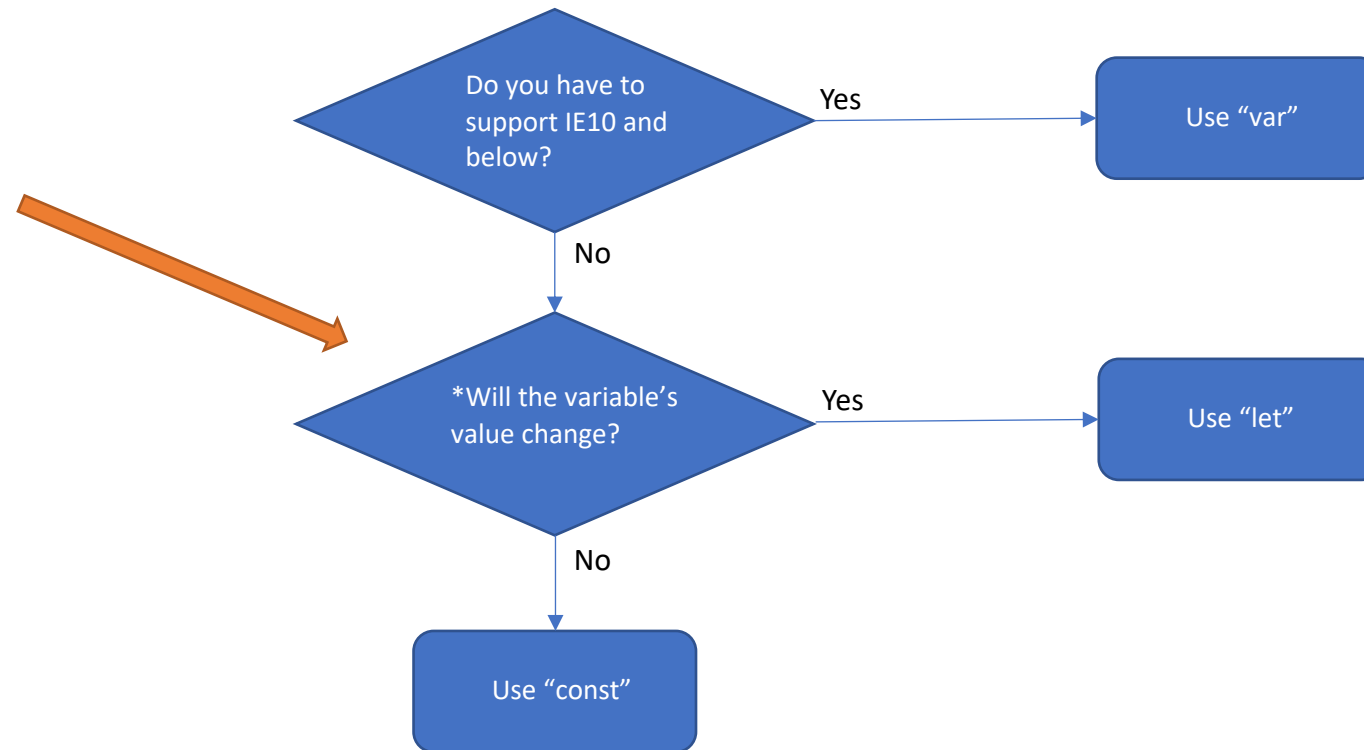
- Variables in JavaScript can be function scoped (var) or block scoped (const or let)
- Unlike some other languages, JavaScript variables declared in a parent function (or the global space) are available inside any child functions

const, let or var?

- Prior to ES6 the only proper way to declare a variable was to use the “var” keyword
- ES6 introduced two new types of variables
 - const
 - These variables can only have their value set once and their core value can not change during the running of your script
 - Variables declared with the “const” keyword are block scoped
 - let
 - Similar to the older “var”, variables declared with the “let” keyword can have their values change during the running of your script
 - Unlike variables declared with the “var” keyword, which are function scoped, variables declared with the “let” keyword are block scoped

Simplified Variable Keyword Decision Tree

* This only applies during each instance of the script running. You would still use a “const” variable if the variable had different values when the script ran at another time.



JavaScript Window Methods

Built-In JavaScript Window Methods

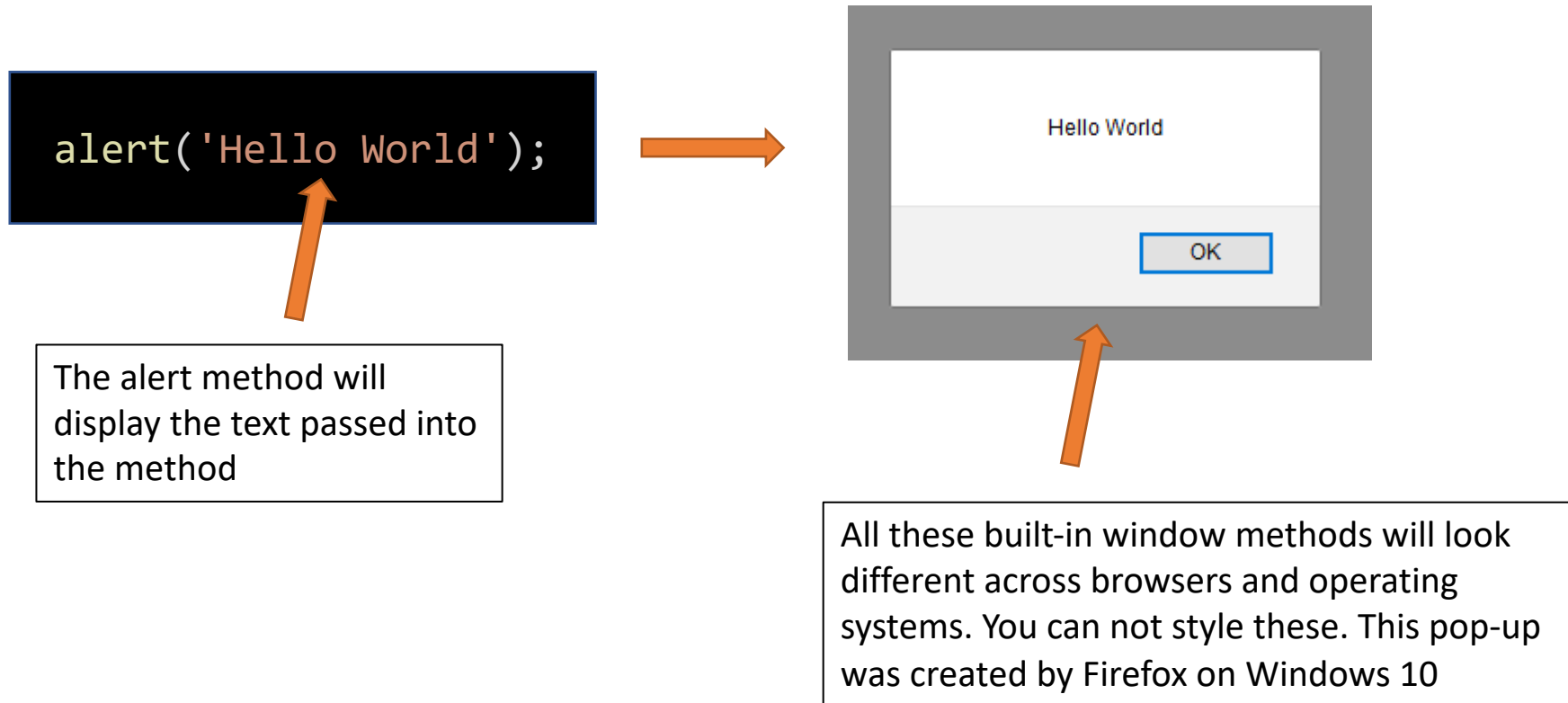
- The built-in window methods "alert", "prompt" and "confirm" are useful for quick prototyping and learning the JavaScript language
- The pop-up windows produced by these methods can not be custom styled or have their functionality changed by the developer
 - If you need a custom pop-up window with custom functionality and style (often called a "modal" window) you will have to create it yourself with HTML/CSS and JavaScript
- Due to the styling and functionality limitations they are not usually used in final production web sites or web applications
- They are mainly used for early prototyping or testing of web sites and applications during development

Where is the Window Object

- The "alert()", "prompt()" and "confirm()" methods belong to the "window" object
 - The "window" object represents the browser window and is built in to all versions of JavaScript that run in the browser
- The "window" object is assumed to be the default object, so writing "window.alert()" (which is still correct) can be shortened to just "alert()"
 - Most developers like to save keystrokes so you will more often see these window methods written without the "window" object before these methods

The alert() Method

- Creates a generic pop-up window that displays the text passed into the method (the text in quotes between the "()")



The prompt() Method

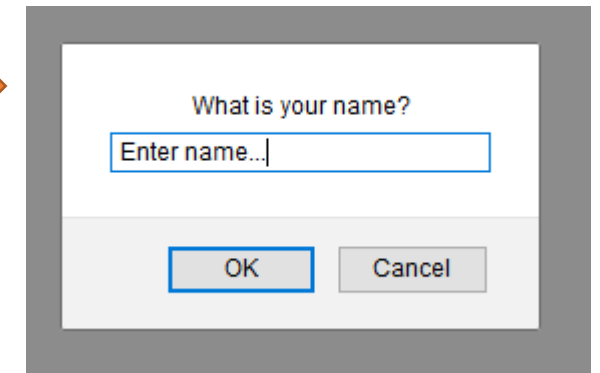
- Creates a generic pop-up window that allows a user to enter text into the pop-up window
- The prompt method "returns" the text that the user entered into the pop-up window

```
const user = prompt('What is your name?', 'Enter name...');
```

The variable we will use to store the text that the user enters into the prompt window

This required parameter represents the text you wish to have displayed in the prompt box

This is an optional parameter and is used to display helper text in the input field




What does "return" mean?

- In many computer languages, methods and functions often process data and then output that processed data
- When a function outputs data it is often done by a return statement
- This is true with built-in methods as well
- We often use a "variable" to capture and store the returned data from a function or method


How to capture and Store Returned Data

- To capture and store returned data from a function or method we can use a variable which has its value set to the function or method call

```
const user = prompt('What is your name?', 'Enter name...');
```



The variable "user" is used to store the data "returned" by the prompt() method



The prompt() method returns text that the user enters into the prompt window

The confirm() Method

- The confirm() method displays a pop-up box that asks the user to "confirm" an action
- The confirm() method returns a special type of data called a "Boolean"
 - A Boolean data type can have two values
 - true
 - false

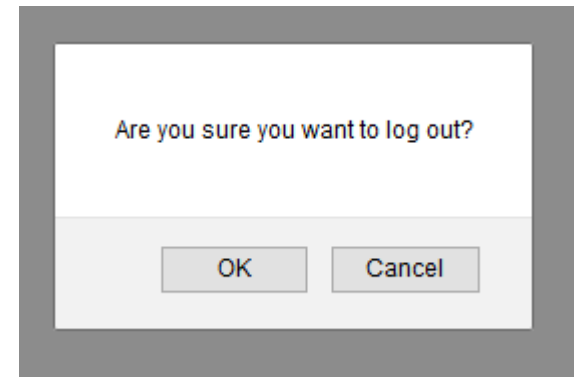
The confirm() Method

```
const logout = confirm('Are you sure you want to log out?');
```

The confirm() method will return a Boolean value (true or false)

The text you want to display within the confirm box

The confirm() method displays pop-up box in the browser that asks the user to confirm an action



Modifying the DOM

- Browsers provide an API (Application Programming Interface) for manipulating the DOM (Document Object Model)
 - DOM API
- Think of the DOM as a representation of the HTML document
 - Each HTML element is a node
 - Each HTML element is an object with several built-in properties and methods
- We will have additional lessons on manipulating the HTML document (the DOM) in a latter lesson

Using innerHTML

- The “innerHTML” property is a property of an HTML element that allows you to set the HTML of an element or get the existing HTML of an element
- The “innerHTML” property is just one of several ways to set or get the HTML of an element

Using innerHTML to Set the HTML

```
<p id="foo">Hello World!</p>
```

We grab the element with the id of "foo" using the "getElementById()" method and then change the "foo" elements HTML content to "Hello Class!" by changing the value of the "foo" elements "innerHTML" property

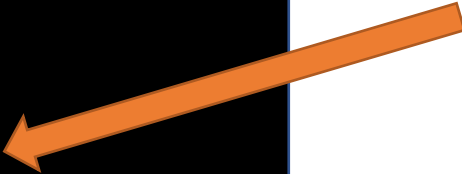
```
document.getElementById('foo').innerHTML = 'Hello Class!';
```

Hello Class!

JavaScript "if" Statements

- The "if" statement runs code between the "{ ... }", if a statement placed between the "(...conditional statement...)" is true

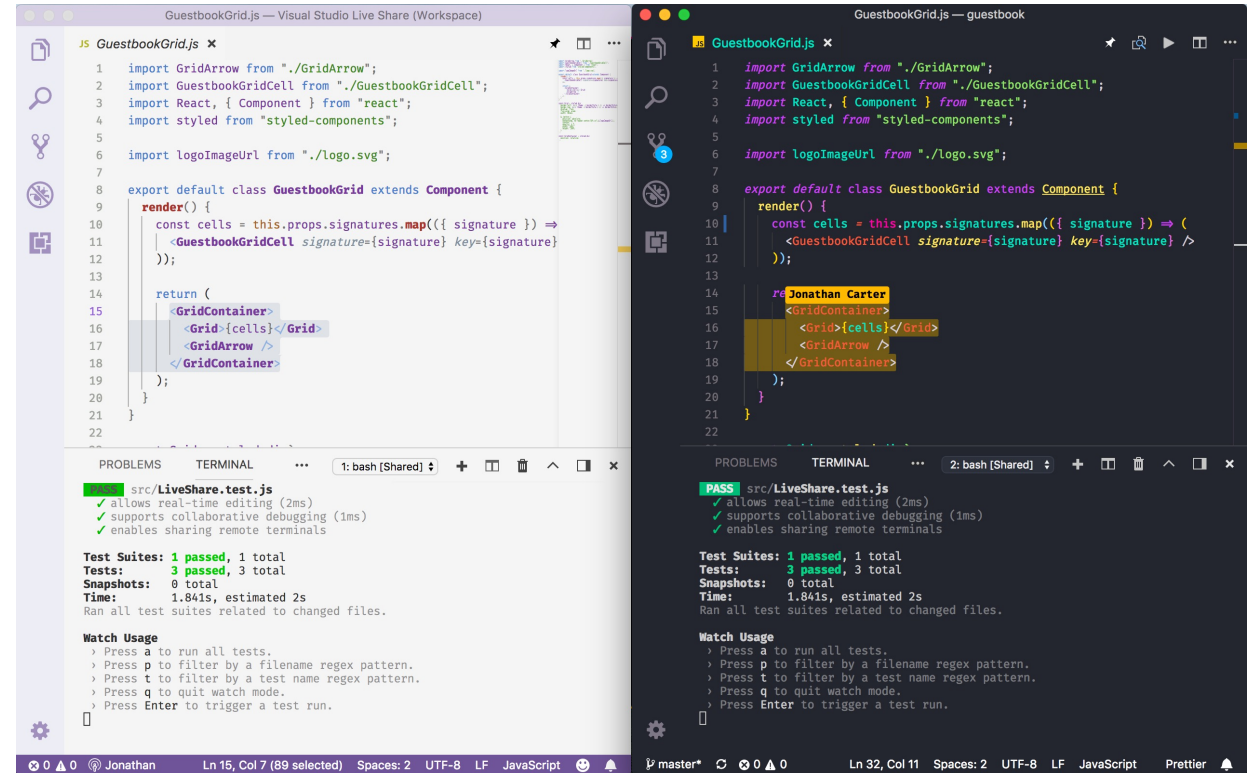
```
const fruit = 'apple';  
  
if(fruit === 'apple'){  
    console.log('I love apples!');  
}
```



This code will only run "if" the "fruit" variable is equal to "apple"

Live Share

- Live share is a Visual Studio code extension that allows multiple people to collaborate on a coding project
- Team members can edit a code simultaneously in real-time and see the edits of their team members in real-time on their computer



Live Share

- To use Live Share you will need to do the following:
 - Install the Live Share extension
 - <https://marketplace.visualstudio.com/items?itemName=MS-vsliveshare.vsliveshare>
 - Create or use an existing Microsoft or GitHub account
 - One team member initiates the share and shares a link with other team members to enable them to join
 - Give this YouTube video a watch for further details
 - <https://youtu.be/A2ceblXTBBc>

