Name ____ Period ______

My First JavaScript Program

Your Tasks (Mark these off as you go)

- ☐ Define key vocabulary
- ☐ Reference a JavaScript program from an HTML page in Visual Studio Code
- ☐ Write code to print to the browser console
- ☐ Add comments to a JavaScript program
- ☐ Receive credit for this lab guide

☐ Define key vocabulary

Computer program

JavaScript statement

Browser console

`<script></script>` tag

***defer* attribute**

Console object

Console method *log*

JavaScript comment

☐ Reference a JavaScript program from an HTML page in Visual Studio Code

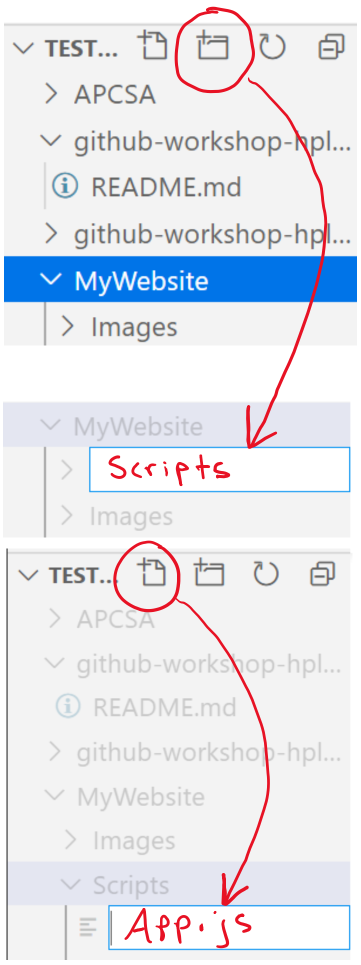
All modern browsers are capable of rendering JavaScript, which makes developing in JavaScript quick and easy. Including JavaScript is a three-step process. First you should create a JavaScript file. Second, you need to create an HTML file. Lastly, you need to reference the JavaScript file from you HTML page using the `<script></script>` tag.

Creating a JavaScript file

JavaScript files end with the **.js* extension and all JavaScript should be written in separate files.

To help you stay organized all JavaScript files should be stored in their own directory. In our example we will call this directory *Scripts*. Below are the steps for creating a new directory and JavaScript file in Visual Studio Code (VSCode).

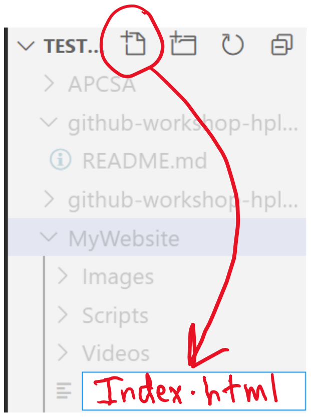
- Select your main website directory. In the example shown right, this is called *MyWebsite*. To create a new directory, click on the new directory icon in the shortcut menu that appears. Call the new directory *Scripts*.
- Now, select the Scripts directory you just created and click on the new file icon in the shortcut menu that appears. Call this new file *App.js*.



Creating an *.html file

Now we need to create an *.html file from which we will reference the App.js file we just created.

- Select your main website directory (*MyWebsite*), then click on the new file icon in the shortcut menu that appears. Call this new file *Index.html*



Reference the JavaScript file from you HTML page using the `<script></script>` tag

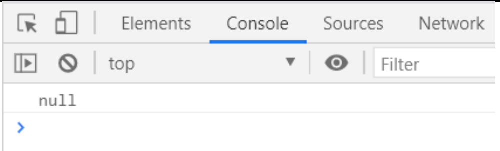
Referencing a JavaScript file from an HTML file requires the `<script></script>` tag. The *src* attribute is used to indicate the location of the JavaScript file. This is illustrated below. But simply referencing a JavaScript file from an HTML file doesn't mean your code will execute as intended.

When a page is loaded in your browser, the code is read from the top down. When the browser encounters a JavaScript file, it will attempt to run the script first, before loading the rest of the page. Consider the following example,

index.html	**App.js**

`<!DOCTYPE html>`
`<html>`
`<head>`
`<script src="Scripts/App.js"></script>`
`</head>`
`<!-- This isn't visible until the script`
`above loads and runs -->`
`<body>`
`  <p>...content after script...</p>`
`</body>`
`</html>`
	`console.log(document.body);`
	
	This code is referencing the `<body></body>` of the page before it has loaded

Output	


Elements Console Sources Network
top Filter
`null`
`>`	

In the above example, the script is loaded in the `<head></head>` section of the webpage – as it should be. But, because it is loaded before the `<body></body>` section, the script cannot reference it and *null* is logged to the console.

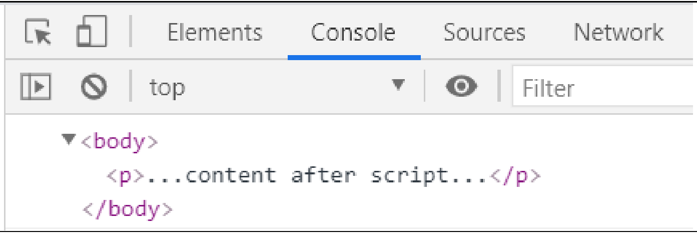
To fix this issue we can use the *defer* attribute. The defer attribute tells the browser not to wait for the script. Instead, the browser will continue to process the HTML. The script loads "in the background", and then runs when the document is fully built.

Here's the same example as above, but with `defer`.

index.html	App.js

`<!DOCTYPE html>`
`<html>`
`<head>`
`<script src="Scripts/App.js" defer></script>`
`</head>`
`<body>`
`  <p>...content after script...</p>`
`</body>`
`</html>`
	`console.log(document.body);`
	this is loaded in the background
	once the page has loaded, the script executes

Output



```
▼<body>
   <p>...content after script...</p>
 </body>
```

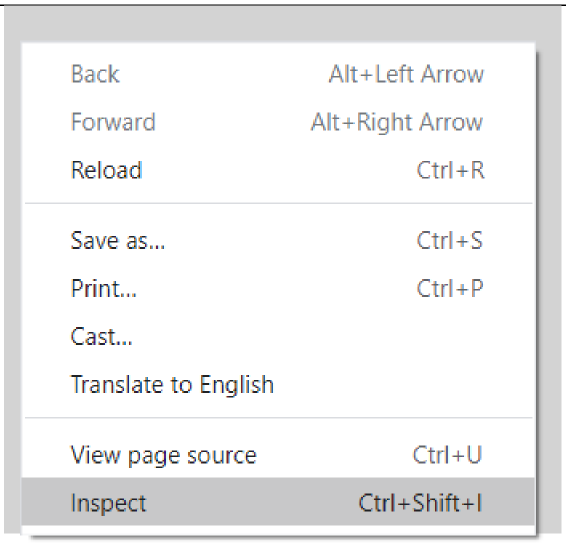
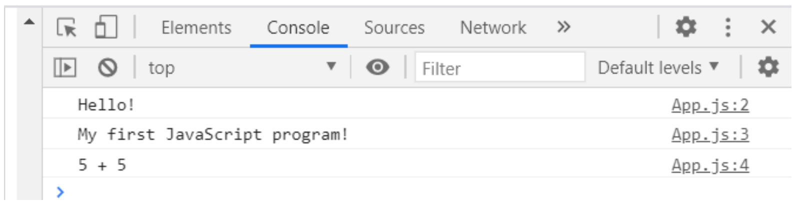
Write code that could be used to reference the *App.js* file from the *Index.html* page. In the `<script></script>` tag be sure to include the *defer* attribute.

File Structure	Code
MyWebsite Index.html App.js	
MyWebsite Index.html **Includes** **Images**: Cat.png **Scripts**: App.js	
MyWebsite **Home**: Index.html **Includes** **Scripts**: App.js	

◻ Write code to print to the browser console

All browsers have consoles that can display important messages, like errors, for developers. If we want to see things that do not appear on the webpage screen, we can print, or log, to the console directly. Using the console to log messages is extremely useful for debugging and interpreting the programs we write.

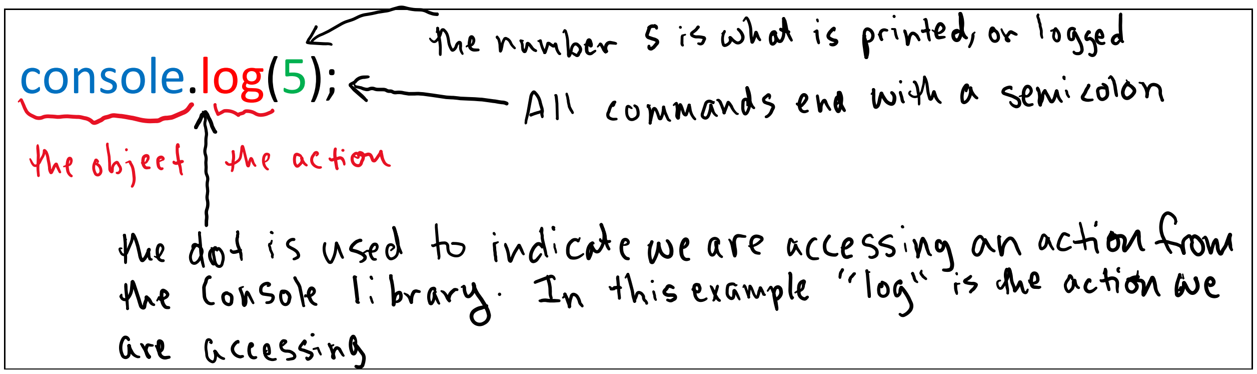
To the view the console from your browser do the following type *Ctrl-Shift-J*, or,

Right click on the webpage where your JavaScript is running and select *inspect*.	 Back Alt+Left Arrow Forward Alt+Right Arrow Reload Ctrl+R Save as... Ctrl+S Print... Ctrl+P Cast... Translate to English View page source Ctrl+U Inspect Ctrl+Shift+I
Select the Console tab	 Elements Console Sources Network top Filter Default levels

```
Hello!                                   App.js:2
My first JavaScript program!             App.js:3
5 + 5                                    App.js:4
```

In JavaScript, the `console` keyword refers to an object. All objects have libraries of built in methods, or actions (more on this later).

One action, or method, that is built into the console object is `log()`. Below is the notation for implementing the `log()` method. The entire statement is referred to as a *command*. When we write `console.log();` what we put inside the parentheses will get printed, or logged, to the console.



```
console.log(5);
```

The number 5 is what is printed, or logged

All commands end with a semicolon

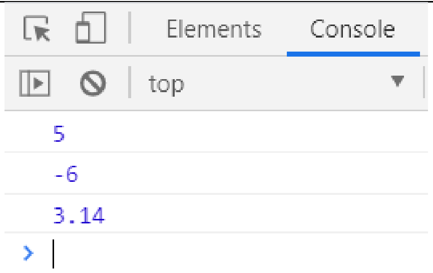
the object

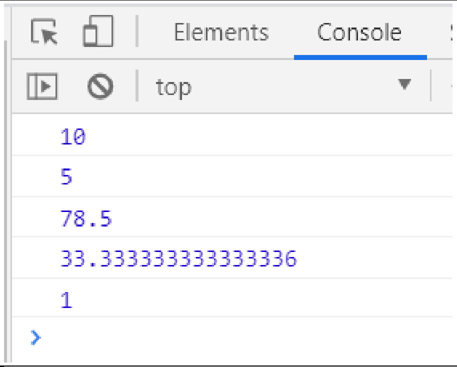
the action

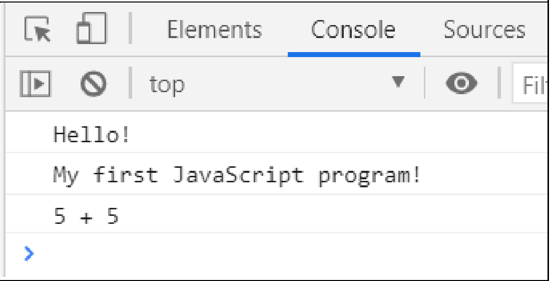
The dot is used to indicate we are accessing an action from the Console library. In this example "log" is the action we are accessing

Numbers of text can be printed to the console. The result of a simple arithmetic operations can also be printed. Some examples are illustrated below,

Statement	Output

`console.log(5);`
`console.log(-6);`
`console.log(3.14);`
	
	`5`
	`-6`
	`3.14`

`console.log(5 + 5);`
`console.log(10 - 5);`
`console.log(3.14*5*5);`
`console.log(100/3);`
`console.log(21%2);`
	
	`10`
	`5`
	`78.5`
	`33.333333333333336`
	`1`

`console.log("Hello!");`
`console.log("My first JavaScript program!"`
`);`
`console.log("5 + 5");`
	
	`Hello!`
	`My first JavaScript program!`
	`5 + 5`

Write three statements that could log the following problems to the console exactly as shown.

```
Problem 1: 79 + 3 * (4 + 82 – 68) -7 + 19
Problem 2: (179 + 21 + 10)/7 + 181
Problem 3: 10389 * 56 * 11 + 2246
```

Write three statements that could log the result of the problems. Let the console do the math!

◻ Add comments to a JavaScript program

As we write JavaScript, we can write comments in our code that the computer will ignore as our program runs. These comments exist just for human readers and are not interpreted as code. There are two types of code comments in JavaScript: single line and multiline

- A single line comment will comment out a single line and is denoted with two forward slashes // preceding it.

```
// Prints 5 to the console
console.log(5);
```

- You can also use a single line comment to comment after a line of code:

```
console.log(5);  // Prints 5
```

- A multi-line comment will comment out multiple lines and is denoted with /* to begin the comment, and */ to end the comment.

```
/*
This is all commented
console.log(10);
None of this is going to run!
console.log(99);
*/
```

- You can also use this syntax to comment something out in the middle of a line of code:

```
console.log(/*IGNORED!*/ 5);  // Still just prints 5
```

Write inline (single line comments) to indicate your name, period, and date
Write a block comment (multi line comment) to indicate what the code below does,

```

console.log("_       _");
console.log("0       0");
console.log("    o");
console.log("   ~~~");
console.log("result = ");
console.log(2*3.14*150);
```

◻ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide.