CSIS 3380: Project / Lab 2:

Project: Display Random Phrases / Flash Cards:

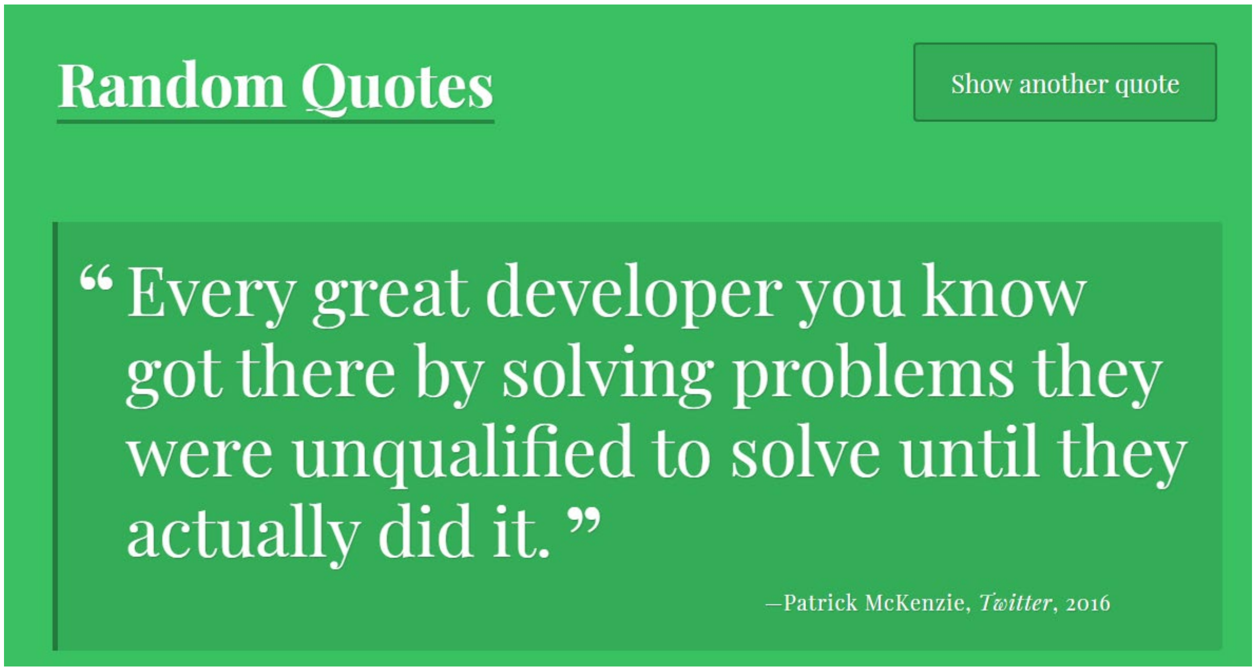
Note: this is an assessment project. Limited help is provided before the submission due date. Questions will be answered after the due date.

This project has 10 marks.

Introduction:

For this first project, you'll create an app that displays random famous quotes each time a button is clicked. You will select your own quotes from famous people. You can use the app for Random Flash Cards or any other idea of your own.

Example:


Random Quotes
Show another quote
“ Every great developer you know got there by solving problems they were unqualified to solve until they actually did it. ”
—Patrick McKenzie, *Twitter*, 2016

You'll locate and select your own quotes. Please select tasteful, positive and uncontroversial quotes, for the sake of this project. You'll use your growing knowledge of basic JavaScript syntax, variables, loops, conditionals and object literals to:

- Build the array of quote objects to store the quotes.
- Write your own functions for selecting random quotes from the array and printing them to the screen.

This project is a fun and effective way for you to practice basic JavaScript skills while also creating a simple interactive portfolio piece to showcase your understanding of JavaScript fundamentals.

The detailed steps of the project is outlined at the end of this document.

This document starts with introduction to some instructions and challenges that help you to accomplish the project.

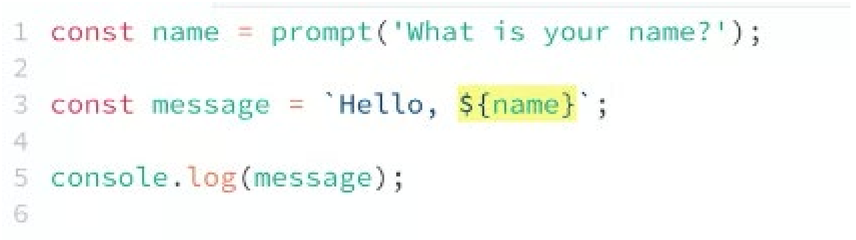
Required Knowledge:

- **BASIC JAVASCRIPT SYNTAX,**
- **VARIABLES, LOOPS,**
- **CONDITIONALS, AND**

Some of the other required information is outlined below:

Template Literals:

Choose folder: start-template literals. Open the file named template-literals.js and update the script tag and index.html, to point to template-literals.js. Writing template literals is almost the same as writing strings with single and double quotes. The most significant differences that you define template literals with backticks instead of quotes. The backtick character is in the upper left-hand corner of the keyboard and it shares a key with the tilde symbol. The existing file create a hello message. Using template literals rewrite the code as shown below and test your code:



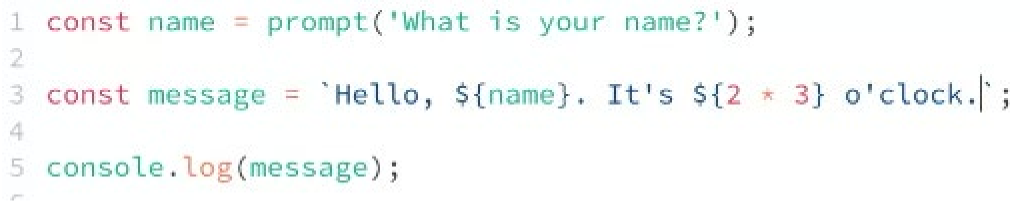
```
const name = prompt('What is your name?');

const message = `Hello, ${name}`;

console.log(message);

```

Also try the following code:



```
const name = prompt('What is your name?');

const message = `Hello, ${name}. It's ${2 * 3} o'clock.`;

console.log(message);
```

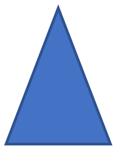
Challenge Question:

Open Challenge1.js and convert it to Template Literal. Create an html body and test your program.

Practice Question:

Go to practice_js_variables_input_output folder and check the practice instructions in the practice.js file. Try to do it yourself.

complete the practice.



Functions:

Do the following exercise to test your knowledge about functions:

1- Create a function named getYear. Do not add any code inside the function yet.
function getYear(){};

2- Inside the function's code block, add the following line of code:
const year = new Date().getFullYear();

This creates a new variable and stores the current year in it. Now, add a statement that returns this variable from the function.

3- Finally, call the getYear function. Store the returned value of the function in a new variable named yearToday.

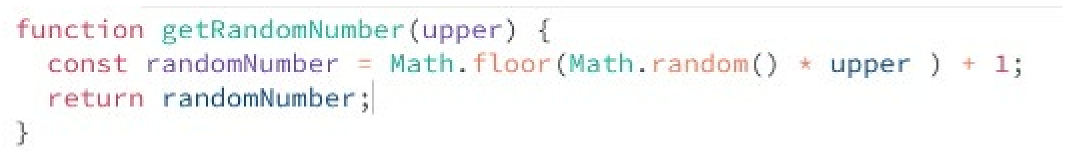
Variable Scope; when using functions

Try the following code, but before testing it guess what you think the following code will display? Why?

```
// Global scope
const person = 'Lee';
function greeting() {
  // Function scope
  const person = 'Meg';
  alert(`Hi, ${person}!`);}
greeting();
alert(`Hi, ${person}!`);
greeting();
```

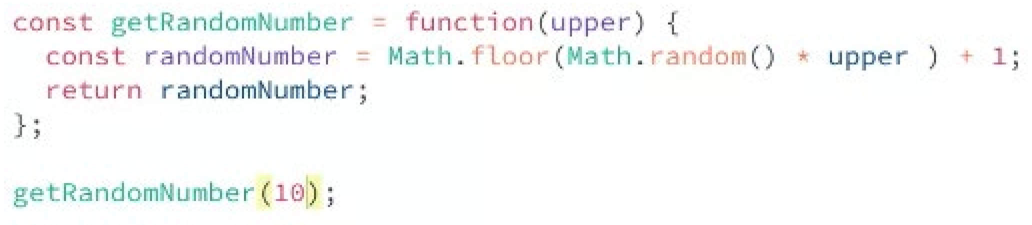
Function Expressions:

So far, you've written functions that look like this, the function keyword, followed by the name of the function. This is what's called a function declaration.



```
function getRandomNumber(upper) {
  const randomNumber = Math.floor(Math.random() * upper ) + 1;
  return randomNumber;
}
```

There's another syntax for creating a function, that's called a function expression. A function expression lets you assign a function to a variable. For example, you could rewrite the getRandomNumber function to look like this.



```
const getRandomNumber = function(upper) {
  const randomNumber = Math.floor(Math.random() * upper ) + 1;
  return randomNumber;
};

getRandomNumber(10);
```

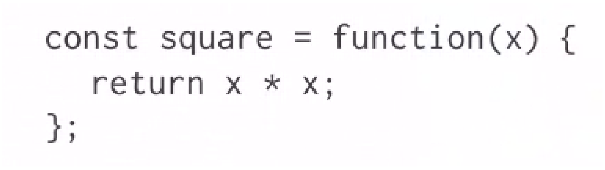
Notice that there's no name after the function keyword. A function without a name after the function keyword is called an **anonymous function**. Instead, the name comes from the variable. In other words, storing a function as a value in a variable named getRandomNumber. Also in this case, you're creating a statement by assigning a value to a variable. You call this type of function the same as you would have function with a name or a function declaration. Use the name of the variable, followed by parenthesis.

You're also going to explore function expressions further in the next stage on arrow functions. Finally, function declrations and function expressions work in much the same way, except for one subtle but important difference. You can call a function declaration before it's defined. This behavior is called hoisting. Function expressions, on the other hand, are not hoisted or lifted up to the top of their scope.

A function expression loads only when the JavaScript engine reaches the line of code it's on. If your program at some point might need to call a function before it's declared, then use function declarations.

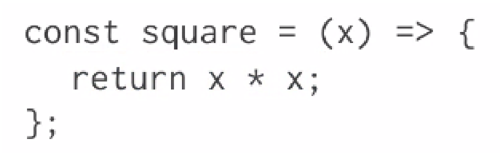
Arrow Functions

You can write a function expression such as:



```
const square = function(x) {
  return x * x;
};
```

As



```
const square = (x) => {
  return x * x;
};
```

Review the supplied document on Concise Arrow Function.pdf

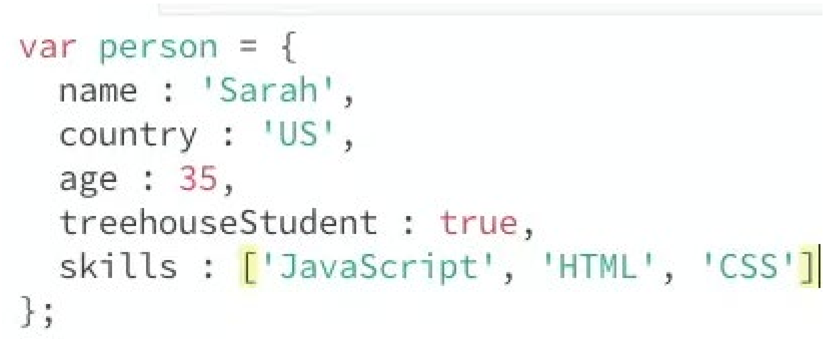
Challenge 2 – Arrow Functions:

Download the data files from “Arrow Function Challenge” and write an arrow function that accept two parameters and generate a random variable between a lower and higher valuem (assume the higher value is 100 if not supplied). Instructions for the challenge are in the .js file.

Work on the program.

Objects in Javascript

The following code shows a simple object in Javascript that could contain any type of variable. Objects in Javascript is organized as Key: value, as can be seen below.



```
var person = {
  name : 'Sarah',
  country : 'US',
  age : 35,
  treehouseStudent : true,
  skills : ['JavaScript', 'HTML', 'CSS']
};
```

Practice Objects, accessing object properties and for_in Syntax

Open “for_in_Object_Syntax” from your data files for further explanation. Notice how we use “(for …. In ….)” and also notice we use person[prop] rather than person.prop. Why do you think the reason is.

“Search Objects” is a small app that demonstrates some of the basic concepts in regards to reading properties of objects.

Project Walkthrough

The instructions below are provided to help. You can add your own code in your approach, but you should only use Javascript and follow the general structure of the app that is provided. Using jQuery or any other library for this project is not allowed. If the following guidelines are not helpful, feel free to use your own code.

To complete this project, follow the instructions below.

Steps:

Download the Project Files

Download the project starter files, unzip them, add them to your project folder. Open the project in your text editor, open the script.js file, load the index.html file in Chrome.

Once unzipped, the file and folder structure of the project directory should not be changed. The files are linked together in a way that depends upon the current directory structure, and making changes could disconnect the CSS and/or JavaScript from the HTML.

The `index.html` file has the basic HTML markup that will be loaded and displayed in the browser. Don't change anything in this file.

The `css` folder contains the styles for the project. You won't need to change anything these files.

The `js` folder contains the `script.js` file. This is where you will write your own JavaScript to control how your app functions and complete the project.

Add data to your quote objects

In this step, you're creating a variable, naming it 'quotes', setting it equal to an empty array, and then adding five or more (as many as your planned cards) empty, comma separated objects to the array.

```
// Empty array
[]

// Empty object
{}
```

Object Properties:

The objects in the `quotes` array store the individual properties of the quotes.

1- Add the following properties to each quote object:

`quote` - *string* - the actual quote

`source` - *string* - the person or character who said it

2- Add a `citation` property to at least one quote object. The value should be a *string* holding a reference to the source of the quote, like the book, movie or song where the quote originates.
3- Add a `year` property to at least one quote object. The value should be a *string* or *number* representing the year the quote originated.

```
// Objects are made up of comma separated pairs of
// properties and values
{
  propertyOne: valueOne,
  propertyTwo: valueTwo,
  propertyThree: valueThree
}
```

Testing your code:

It's important to test your code frequently to catch errors as they arise, which can help prevent having to track them down later. Now that you have your array of quote objects, use the console.log() method to log out the `quotes` variable to see your array of quotes in the browser.

The getRandomQuote Function:

The `getRandomQuote` function should create a random number and use that random number to return a random quote object from the `quotes` array.

This function needs to accomplish three tasks: use `Math.floor`, `Math.random` and the length of the quotes array to generate a random number, use bracket notation and the random number variable to grab a quote object from the quotes array, and lastly, return the random quote object.

- Create a function named `getRandomQuote`.
- In the function body, create a variable to store a random number ranging from zero to the index of the last item in the `quotes` array.
- Lastly, the function should return a random quote object using the random number variable above and bracket notation on the `quotes` array.

To help you get started, you can copy the following code snippet directly into your `script.js` file and use the code comments to help guide you forward.

```
function getRandomQuote() {
  // 1. Create a variable that generates a random number
  // between zero and the last index in the `quotes` array

  // 2. Use the random number variable and bracket notation
  // to grab a random object from the `quotes` array, and
  // store it in a variable

  // 3. Return the variable storing the random quote object
}
```

The printQuote Function:

Your app should display a new quote every time the user clicks the "Show another quote" button. At the bottom of the `js/script.js` file, we included a line of code that calls the `printQuote` function each time a click happens.

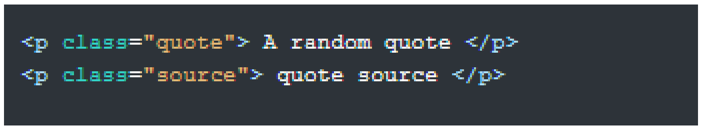
So when the `printQuote` function is called, it needs to get a random quote object from the quotes array, use that random quote object to assemble a string of HTML and quote properties, and then update the HTML to include that string.

1. Create a function named `printQuote`.
2. You will program the `printQuote` function to perform three tasks: call the `getRandomQuote` function, use the returned quote object to build a string of HTML and quote properties, then use that string to display a random quote in the browser.
3. In the body of the `printQuote` function, create a variable to store a random quote object from the `getRandomQuote()` function.
   To get a random quote object, create a variable and set it equal to a call to the `getRandomQuote()` function.

In script.js
Type:

```
let RandomQuote = getRandomQuote()
```

4. Create another variable to store the HTML string. Set it equal to a string containing two `<p>` elements. Use this code snippet as a guide for what the HTML string should look like at this point:



```
<p class="quote"> A random quote </p>
<p class="source"> quote source </p>
```

- The first `<p>` element should have a class equal to “quote”, and the random quote object’s `.quote` property nested between the opening and closing `<p>` tags.
- The second `<p>` element should have a class equal to “source”, and the random quote object’s `.source` property nested between the tags.

In script.js
Type:

```
let PtoHTML1 = “<p class=’source’>” + randomQuote.quote + “<p class=’source’>” +
randomQuote.source;
```

- Leave off the second closing </p> tag for now. That will get concatenated later.

Create two separate if statements below the variables

What makes this next p art of the project particularly tricky is that this function needs to make decisions and include some of the quote properties only if they exist. To do this, create two `if` statements below the two variables you've already created in the `printQuote` function.

```
// Basic if statement
if (/* your condition */) {
  // Code to run if your condition evaluates to true
}
```

If the quote property exists, add it to the string you started above using the required format. In the code block of each `if` statement concatenate a `<span></span>` element with the appropriate class names: "citation" for the `citation` property and "year" for the `year` property. And nest the `randomQuote.citation` and `randomQuote.year` properties between their respective `<span>` tags.

After the `if` statements, concatenate the closing `</p>` tag to your HTML string. When complete, the string should resemble something like this if it has both the `citation` and `year` properties.

```
p class="quote">quote text</p>
<p class="source">quote source
  <span class="citation">quote citation</span>
  <span class="year">quote year</span>
</p>
```

Without the `citation` and `year` properties, the string should resemble something like this.

```
<p class="quote">quote text</p>
<p class="source">quote source</p>
```

And lastly, so your string can be printed to the page, insert it into the HTML like so:

```
document.getElementById('quote-box').innerHTML = yourStringHere;
```

To help you get started, you can copy the following code snippet directly into your `script.js` file and use the code comments to help guide you forward.

```
function printQuote() {
 // 1. Create a variable that calls the getRandomQuote()
 // function

 // 2. Create a variable that initiates your HTML string with
 // the first two <p></p> elements, their classNames,
 // and the quote and source properties, but leave off
 // the second closing `</p>` tag for now

 // 3. Use an if statement to check if the citation property
 // exists, and if it does, concatenate a <span></span>
 // element, appropriate className, and citation property
 // to the HTML string

 // 4. Use an if statement to check of the year property exists,
 // and if it does, concatenate a <span></span> element,
 // appropriate className, and year property to the HTML
 //string

 // 5. After the two if statements, concatenate the closing </p>
 // tag to the HTML string

 // 6. set the innerHTML of the quote-box div to equal the
 // complete HTML string}
```

Final code test:

Inside the `printQuote` function, log out the random quote object, one or more of the properties from the random quote object, and lastly, at the end of the function, the complete HTML string. Clicking the "Show another quote" button should run the code in the `printQuote` function, including any log statements you add.

Code Comments:

Before you submit, be sure to add your own code comments that briefly describe what your code is doing to make it easier for other developers and your future self to understand at a glance.

Finishing up the Project and Before you Submit

Zip all the files in one folder. Make sure when it is unzipped, all resources are properly linked. I will run the HTML code and expect it to run other wise there will be no mark.

Enjoy Coding!