NM2207
Session 04
Challenges

Before attempting the challenges, you are expected to have watched and coded along with the Lecture videos. A tutorial is meant to practice the skills presented in the video lecture, and show you more applications of it. Tutors will explain the challenges and answer the questions you may have**.**

The challenges are due to be completed at the end of class each week for full credit which is also attendance. Submitting by midnight of the same day (ie Thursday night) accounts for half the credit.

Overview of what we will do today:

- Practice creating a lot of functions
- We will add a click counter to the header panel of the web page.
- Then we'll add a slider to the aside panel that will change the header panel background colour between black and white (and shades of grey in between).

Part 1

Summary of learnings

- Revising functions and DOM manipulation
- Using inbuilt event handlers of input elements

Warm up (20 minutes)

1. In your appscripts/main.js file within the Session04.class folder, write a function. This function returns no value.
    a. Add the following steps/prompts (b-f) as comments to your main.js, and paste the code corresponding to them under each prompt.
    b. Edit your empty function so that it prints “Hello” to the console.log.
    c. Add a function call so that your function is executed once, each time the page is loaded.
    d. Now, comment out the command that satisfies Step b. Instead of printing to the console log, edit your function so that it should print “Hello” to the header element.
    e. Declare a variable called counter. And initialize it to 0, in the first few lines of your main.js file, before you define the function. Here is an example of how declaration and initialization can be different steps.

```
// Declare at the beginning
var firstName, lastName, price, discount, fullPrice;

// Use later
firstName = "John";
lastName = "Doe";

price = 19.90;
discount = 0.10;
```

f. Comment out the command that Satisfies step d. Now, instead of printing “Hello”, your function should now print counter’s value to the header element.
    i. This involves a document.getElementById() command and then setting the innerHTML property. Do you remember how to do this? Check the Table in the Session03_challenges document if you need to refresh your memory.

Using the inbuilt event listeners of form elements (45 minutes)

- *From the Session 2 assignment and this week’s video lectures, we know about <input> elements, and how we can use onclick to do something, such as generate an alert or post a form data.*
- *“onclick” and other “onsomething” attributes of <input> elements are known as event handlers. They handle what to do when a user does something to a particular element, such as click it. But they can also handle general user actions which are not specific to an element, such as when a user scrolls or hovers over the close button.*
- *Any action that a user performs is called an event.*
- *The process of setting up an event handler is known as adding an event listener. When the event happens, the listener function or command will be triggered.*

2. In your html file, create a button in the center of the footer and set its onclick property to the function you created in Step 1. Now, this function can be executed more than once. In fact, it will be executed each time the button is clicked!

- Comment out the function call of Step 1c. Now the function will only be executed on a click and not each time the page is loaded.

3. Go back to your main.js. Copy the following prompts into your function as comments, and then fill it up with code to correspond to each step. Here’s what it should do:

- Track the number of clicks. It can do this by increasing the value of the counter variable that you set in Step 1e, by 1,
- Changes the text in the header element to read something like "OK, I have now received X clicks" (where X is the click count). So comment out the command for Step 1f and format it like this instead.
- Now make it lie: have it print out 100 times the number of actual clicks it has received.

4. Save your progress and create a backup version of your main.js file, because we plan to break things in Part 2.

(Suggested 10-minute break)

Part 2

Summary of learnings

Variable scope (the difference between variables declared inside function and those declared outside functions)

- ***Variable Declaration:*** *Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword or the let keyword.*
- ***Variable Initialization:*** *This usually occurs when a variable is declared. Here the variable is assigned a memory or space by the JavaScript engine. Because of this, once a variable is declared, it takes a value of undefined even before assignment.*
- ***Variable Assignment:*** *Variable assignment is usually the most important step when using a variable. Here the variable is assigned data which is a value using the assignment operator "=". Values in JavaScript take one of the standard JavaScript datatypes (string, number, boolean, null, undefined)*

Reference: https://www.tutorialspoint.com/javascript/javascript_variables.htm

Long-term and short-term memory loss with variable scope (20 minutes)

4. Let's play with a new code snippet. It is provided as part2.js in the appscripts folder. Notice the pre-function block of statements before the function, the function block statements within the function, and the post-function block. You do not need to paste this code anywhere, as it is already linked to your index.html through a line within the head element. You should now uncomment this line in index.html.

```
//PRE-FUNCTION BLOCK
var counter2; //LINE 1: declaring a variable. Current value = undefined
counter2 = 5; //LINE 2: assigning a variable. Current value = 5


function foo() {
//FUNCTION BLOCK
        counter2 = counter2 + 1;
   console.log("Inside the function block: counter2's value is:" + counter2);

}


//POST-FUNCTION BLOCK
console.log("Post-function block: counter2's value is:" + counter2);
foo();
```

Now, to understand variable scope, we will edit part2.js. Shuffle Line 1 and Line 2 between different positions in the pre-, post- and the function block. Remember that Line 1 should always occur before Line 2!

In other words, move your declaration and assignment of counter2 so that you try out all the conditions in the Table 1.

Please fill up the following table with your findings. Note the ORDER in which console.log statements are printed. Why is this so?

Table 1

	Output on the console	**Why**
Line 1 and Line 2 are in current position		
Line 2 is inside function		

block BEFORE "counter2 = counter2+1" expression		
Line 2 is inside function block AFTER "counter2 = counter2+1" expression		
Line 2 is in post-function block, after function call		

Part 3

Summary of learnings

- Formatting color strings
- More practice with variable scope
- The Math function

Color strings and more event handlers (30 minutes)

5. Now back in your index.html file, add a slider to the aside element.
6. Set attributes min, max (don't foget to quote the values) check your slider - how does it move?
7. Now set the step attribute to .01
   Now how does it move?

8. Set the 'onchange' event handler of the slider to a new function name, and add that as an empty function in your main.js file.
   a. Make it print something to the console log.
   b. Save, reload, and test to make sure that the function is called each time you change the value of the slider.
9. Now, comment out the console.log statement for Step 8b. Define this function:
   a. set the style.backgroundColor of the header element to be black (when the slider is 0), white (when the slider is 1), and shades of grey for values in between. Use the rgb() method of specifying the color.
   b. hint: "rgb(255,255,255)" would specify white
   c. another hint: you have to use integers (e.g. 45) for the color values. Floats (eg 45.3) won't work.
   d. yet another hint: 'Math' is a predefined object with lots of handy methods. One of them is 'floor' that takes a floating point number and returns the integer part.
      Thus Math.floor(3.1416) is equal to 3.

Bonus

Bonus questions are not compulsory and are not graded. If you like, you can attempt them as they offer a chance to do some more practice.

10. In step 8, instead of setting the "onchange" event handler, use the "input" event handler instead. How does that change the behavior?
11. What is the scope of slider.value? Currently we refer to it inside an event listener (the function we attached through the event handler) for the slider. Could we access it inside the event listener for the button? Why/Why not? Test it out by printing the value of the slider to the console window from inside the button click function.
12. Set another function to the 'onclick' event handler of the slider.

    a. This function should print a message to the console with the slider value (the 'value' attribute of the slider).

13. Create 2 radio buttons on the nav for selecting between 'Option1' and 'Option2' Note: giving the two radio button element the same value for the 'name' attribute is all you need to do to make sure only one button is selected at a time.
14. Give each radio button a listener for the 'onchange' event with a function that does something ridiculous to some aspect of your web page. (For example, you could switch the background images between the body and the aside; you could change the behavior of the slider; etc)

PSA

Inbuilt event handlers are not a part of best practice for web programming. Next week, we will stop using them because we want to separate the JavaScript out of the html page completely. Why? See https://www.thoughtco.com/moving-javascript-out-of-the-web-page-2037542

So next week, we will be doing this, and you can start reading about it already: https://www.w3schools.com/js/js_htmldom_eventlistener.asp

Reflection

- What kind of app would you like to make with skills you expect to acquire in this course?
- What might you use sliders to do in that app?
- What might you use radio buttons to do in that app?
- Can you imagine roughly what your code would look like for using these interface elements in your app?