Assignment 4 (Part 1): Hands-on Exercises in Class

Note that these exercises are expected to be completed open book. Furthermore, if you experience difficulty with answering any of the questions, you may discuss with your colleagues as well as seek assistance from the instructor and/or the Teaching Assistant (TA), in order to successfully resolve such difficulty. These exercises are simply meant to provide you with some guidance on how you should think about providing your solution to Assignment 4. The objectives of the exercises are to enable you to get started working on as well as to assist you with completing Assignment 4. Hopefully, this objective will be accomplished by the time you are done completing the exercises.

All points, for the various tasks that you are required to complete as well as questions that you are required to answer in these exercises, will have equal weights.

You may use any text editor of your choice to complete these exercises.

(1.) Open a text editor of your choice, create a blank file using this text editor and name the file ***practice-script.js***.

(2.) Using any text editor of your choice, write the HTML code for an HTML document that has only one empty element in it. (***Hint: You will want to include the <!DOCTYPE >, <html></html>, <head></head> and <body></body> elements in the HTML code that you write here. Furthermore, you will want the <head></head> and <body></body> elements to be nested inside the <html></html> element as well as you will want the element to be nested inside the <body></body> element.***)

(3.) Give the empty element, which you created in the step number 2 above, an ***id*** value of your choice.

(4.) Apply JavaScript to the HTML document, which you created in the step number 2 above, by using the separate, external file that is named ***practice-script.js***, which you created in the step number 1 above.

(5.) In the ***practice-script.js*** file, write JavaScript code that inserts/adds the following content to the empty element, which you created in the step number 2 above:

Hello World!
Hello World!
Hello World!

When a browser displays your HTML document, there should be three separate lines of the "Hello World!" text. Hint: Your JavaScript code should do the following:

(A.) In the ***practice-script.js*** file, create a string variable that holds/stores the three separate lines of the "***Hello World!***" text. You can name this variable with any identifier (i.e., variable name) of your choice. This string could look like this: "Hello World!
 Hello World!
 Hello World!" (note that this string simply contains HTML code that you could have manually written yourself to create these three separate lines of the "***Hello World!***" text; by using JavaScript to create these three separate lines of the "***Hello World!***" text, you are now programmatically creating HTML code, rather than manually writing this HTML code yourself).

(B.) In the ***practice-script.js*** file, use the ***id***, of the element in your HTML file, to obtain an object that represents the element.

(C.) In the ***practice-script.js*** file, use this object, which represents the element, to assign the following value to the ***innerHTML*** property of this object: The value held/stored in the string variable that you created in the step number 5A above.

Note this technique of programmatically creating HTML code as a string and assigning the value of this string to the ***innerHTML*** property of an object. This technique is often used to notify a form user of any invalid

data that they might have entered into a form. The object, whose ***innerHTML*** property this string value is assigned to, is usually an object that represents an HTML element (e.g., a element, a element, etc.) in which the error message is displayed to the form user.

Two of the files included in your Assignment 4 folder are named ***index.html*** and ***scripts.js***. Open this HTML file and this JavaScript file in a text editor of your choice.

(6.) Complete Table 1 below, by providing the requested information about the various Graphical User Interface (GUI) features, in the form in the ***index.html*** file, which are used to enable the form user to provide data into this form.

Table 1: GUI Features in the *index.html* File that are Used to Provide Form Input.

Input Field Displayed by Browser	Element in HTML Code that Browser Uses to Display Input Field	Id of Element in HTML Code that Browser Uses to Display Input Field
Name Field		
Username Field		
Password Field		
Age Field		
Short Bio Field		
Gender Radio Buttons		
Dog Liking Certification Check Box		
Favorite Dog Breed Drop Down List		

(7.) Radio buttons are supposed to enable a form user to make a choice out of a set of mutually exclusive choices. Radio buttons that form a set of mutually exclusive choices are called a Radio Group. A Radio Group is created when all the radio buttons in the group are assigned the same value for their ***name*** attributes.

(A.) Play around with the four gender radio buttons on the form in the ***index.html*** file and answer the question whether all these gender radio buttons are in the same Radio Group or not; i.e., whether or not these four gender radio buttons enable a form user to make a choice out of a set of mutually exclusive gender choices. (***Hint: Very carefully scrutinize the values assigned to the name attributes of all four gender radio buttons and determine whether all these values are EXACTLY the same.***)

(B.) If you conclude that all four gender radio buttons are ***NOT*** in the same Radio Group, how would you make them all to be in the same Radio Group?

Recall the following major steps that a programmer needs to perform, in order to implement form validation:

(i.) Determine the values entered into the form.
(ii.) Determine whether the values entered into the form meet certain specified requirements.
(iii.) Notify the form user that they have entered wrong values in the form.

In the next few exercises, you will work on these major three steps that a programmer needs to perform, in order to implement form validation. Notice that the JavaScript code in the ***scripts.js*** file has already been applied to the HTML code in the ***index.html*** file (you already opened these two files, just before completing Step 6 above).

(8.) Add the ***action*** attribute to the <form></form> element in the ***index.html*** file and assign the value of "***`javascript:validateForm()`***" to this ***action*** attribute. By assigning this value to this attribute, when the submit button on the form in the ***index.html*** file is clicked, the data entered into this form will be sent to the ***`validateForm()`*** function in the JavaScript code in the ***scripts.js*** file.

(9.) **Determine the values entered into the form.**

(A.) Check out lines 3 through 10 in the ***scripts.js*** file and take a look at the incomplete JavaScript code on these lines (this code includes JavaScript statements that retrieve the values that the form user entered into the form and submitted).
(B.) Complete the JavaScript code on lines 3 through 10 in the ***scripts.js*** file.

(10.) **Determine whether the values entered into the form meet certain specified requirements.**

(A.) Check out lines 14 through 18 in the ***scripts.js*** file and take a look at the pseudocode on these lines (this pseudocode determines whether the values entered in the form meet certain specified requirements).
(B.) When writing JavaScript code to replace the pseudocode on lines 14 through 18 in the ***scripts.js*** file, how many selection branches (i.e., ***if*** statement branches) will you require to test the validity of all the values entered into the form that need to be validated? (***Hint: There isn't only one correct answer to this question, but you certainly can't validate all the values entered into the form that need to be validated using only the one selection branch or if statement branch shown in this pseudocode.***)

(11.) **Notify the form user that they have entered wrong values in the form.**

(A.) Check out lines 22 through 66 in the ***scripts.js*** file and take a look at the pseudocode on these lines (this pseudocode builds a long string that contains the notification that you will display to a form user, whether they entered valid or invalid data into the form).
(B.) Complete the JavaScript code on lines 24, 25, 41 and 58 in the ***scripts.js*** file.

(12.) The function, on lines 68 through 82 in the ***scripts.js*** file, ensures that the "***I did not say I like dogs***" choice is the only favorite dog choice selected whenever a form user unchecks the "***I certify that I like dogs***" check box. Complete line 85 in the ***scripts.js*** file, by writing JavaScript code that adds an event listener to the `selectFavoriteDogBreed()` function.