JavaScript Assessment

This assessment includes two exercises. Follow the instructions for each exercise carefully to complete the assessment. You have one hour to complete the assessment.

Exercise1

1. Download javascript_assessment_exercise_1.zip which is included with this assessment on moodle.
2. Unzip this file to an empty folder on your device.
3. Note there are two files: error.html and in js folder: error.js.
4. Open error.html in a web browser and check the console for error message.
5. Please add a comment to js/error.js describing the error reported.
6. Note the comments, in js/error.js. Please modify this file to include javascript to handle the error reported.
7. Save all changes and test solution by reloading error.html in the browser and checking there is no error reported in the console.

   NOTE: AT THIS POINT SAVE ALL CHANGES AND TEST SOLUTION IS WORKING AS EXPECTED. WHEN YOU CONFIRM SOLUTION IS WORKING AS EXPECTED, THEN ZIP UP THE HTML AND JS FILES MODIFIED FOR EXERCISE 1 (name it exercise_1.zip) AND EMAIL THIS FILE to kevin.obrien@gmit.ie

Exercise 2

1. Download javascript_assessment_exercise_2.zip which is included with this assessment on moodle.
2. Unzip this file to an empty folder on your device.
3. Note there are two files: addNames.html and nameList.html. In the same folder as these two files is a folder named js.
4. In the js folder, the following three javascript files exist:
   a. addNames.js
   b. nameList.js
   c. clickEvent.js

   These files include some comments and function names to help you with this lab exercise.

5. Note that in addNames.html there are two buttons – Add and Clear Storage. As part of this exercise the localStorage object will be used. For the Clear Storage button note that in the addNames.html there is a function named clear_storage(). Please create this function in the addNames.js file. This function should clear the localStorage object of all data.
6. In addNames.js, create a names object constructor function, so that many names objects can be created for first name and surname. Call this function names. Don't forget the function arguments.
7. For the Add button in addNames.html there is a function named add(). Please create this function in the addNames.js file.
8. The add() function should do the following:

   a. Create a unique key value, which will be used for storing the object instance. You could use a static string appended to a unique value such as the storage length property for this purpose.
   b. Invoke the names function to read the values entered in the text fields in addNames.html to create an instance of the names object.
   c. Convert this instance of the names object to JSON using the JSON objects stringify method (JSON.stringify()).
   d. Use the localStorage objects setItem method to add the key, as defined in step a, and the value, as defined in step c, to local storage.
   e. Add the following line of code as the last line in the function to reload the page so that the web page is refreshed:

      ```
      window.location.reload();
      ```

9. Save all changes and open the addNames.html in a web browser – preferably chrome or firefox.
10. Type some text into the text fields and click the Add button. Then type different text into the texts field and click the Add button. This should ensure you have two keys, with JSON formatted values saved to the local web storage.
11. In nameList.html note that the body has an id attribute set to “namesListBody”.
12. In nameList.js add an event listener on the body element which handles a load event.
13. The event listener should include a function name, which requires no parameters. This function name should be named populate_page and the event flow used in the event listener definition should be bubble.
    Note: If using firefox, then you will need to include an argument for the event object in the populate_page function, so will need to define the function as an anonymous function in the event listener definition. Within the function then refer to the argument for event object to check its properties and invoke its methods.
14. In nameList.js create the function populate_page().
15. This function should do the following:
    a. Build a html string which creates a table definition with rows and columns.
    b. The table definition should begin as follows:

    ```
    <table border='1px solid black' id='t1'><tr><td>First Name</td><td>Surname</td><tr>
    ```

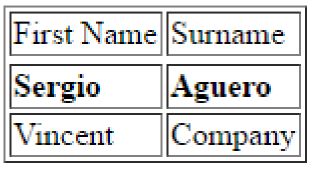
    c. Subsequent rows, must include the firstname and surname stored in the localStorage object. Building up the string with this data should be done using a loop. Please refer to the Javascript Web Storage Lab 1, which includes a solution for reading data from the localStorage object. Note the use of the JSON objects parse method.
    d. Please ensure the first row in the table with data from the localStorage object has its text set to bold. Wrap this text in the <b> tags. E.g:

    ```
    <td><b>John</b></td><td><b>Smith</b></td>
    ```

    e. Subsequent rows should not have the text set to bold.
    f. Note that the web page nameList.html includes a <div> element whose id is set to “nameList”.
    g. Target this element and assign the string with the html table definition to it – you can use innerHTML for this.

h. Save all changes to nameList.js

16. Open addNames.html, click on the "Go To Name List" button. You should see the two names added in step 10 displayed in a table on nameList.html. For example, you should see the following:



First Name	Surname
Sergio	**Aguero**
Vincent	Company

Note: First record in table is highlighted in bold.

17. Save all changes to nameList.html.
18. In js/clickEvent.js, add an event listener on the <table> element which handles a click event. Note in step 15 b, the table element includes an id of "t1".
19. The event listener should include a function named removeItem which requires no parameters. The event flow used in the event listener definition should be bubble.
20. The removeItem function should remove the full row when a row in the table on the web page – nameList.html is clicked. Refer to JavaScript Events Lab 4 for an example of how this is done. Note, that parent nodes need to be targeted, and from the table target the row for deletion using the removeChild method, as shown in JavaScript Events Lab 4. No need to implement backwards capability so that function will work on IE versions 5 to 8. See event-delegation-solution_2.zip for example of where backwards capability is not implemented. Use this solution as a reference.
21. The removeItem function should include a call to the event objects stopPropagation method.

NOTE: AT THIS POINT SAVE ALL CHANGES AND TEST SOLUTION IS WORKING AS EXPECTED. WHEN YOU CONFIRM SOLUTION IS WORKING AS EXPECTED, THEN ZIP UP ALL THE HTML AND JS FILES MODIFIED FOR EXERCISE 2 (name it exercise_2.zip) AND EMAIL THIS FILE to kevin.obrien@gmit.ie