SWE 642: Assignment 2

JavaScript, Cookie, Ajax, & Bootstrap

This assignment is an extension of the previous Assignment and requires you to enhance the page using JavaScript and Bootstrap. You will have an opportunity to demonstrate how to use cookie, JavaScript functions, event handling, Bootstrap, and Ajax implementation. The following are specific details of the assignment. You will continue to use Amazon S3 with your free-tier AWS account to host this homework. You may create another S3 bucket or another folder in the existing bucket. Reference HW1 for setting up a static website on Amazon S3

- **Navigation Bar** – Redesign your homework 1 solution to use Bootstrap navigation bar with a medium to dark background. The idea is that when you click a tab on the top of the page, it should swap the content of the page. https://getbootstrap.com/docs/5.1/components/navbar/

- **Cookie and Greetings Implementation** – On the Student Survey Form Page, use a **cookie** to hold a user's information and display a greeting with the user's name from the cookie. The greeting displayed should be an appropriate greeting based on the time of the day (e.g., good morning, good afternoon, or good evening) when a user visits the web site. (An example of such greeting is "Good Afternoon John, welcome to SWE642 Survey" when the user John visits your page in the afternoon).
    - If the name stored in the cookie is different from the user opening your page, have a hyperlink that prompts the user to enter his/her name. Use the correct name to reset the document's cookie. You may need to reload the page (automatically) to get the new name after resetting the cookie. Set an expiration time for the cookie.
    - If no cookie has been set, then prompt the user to enter his/her name and set the cookie.

- **Survey Form Extension and Average & Maximum Computation – Add an additional text box called "Data", and two output fields labeled Average and Maximum on the Student Survey Form.**
    - A user is required to enter ten comma separated numbers ranging from 1 through100 in the Data field.
    - Create JavaScript function(s) that will calculate Average and Maximum of the numbers entered in the Data field.
    - When the user is finished typing ten numbers and moves out of the Data field, the system should automatically compute the average and maximum of the ten numbers entered in the data field and populate Average and Maximum output fields.
    - Make sure the Data field contains at least ten comma separated values between 1 and 100 (inclusive), else print an error message.

- **Form Validation Event Handling – Develop an event handler to handle onclick event when a user clicks the submit button on the Survey Form. The event handler should validate the following:**

    - The Name text box should contain only Alphabets.

- The Address text boxes should contain only appropriate numeric, alphabet or alphanumeric characters.
- Make sure at least two checkboxes are checked.
- Make sure a radio button option is selected. ○ The Email address format should be valid.

Validate above requirements and alert the user with a consolidated error message if the user missed filling out anything on the form that did not meet above criteria. Your validation warning errors can be implemented using window's dialog box.
Only the fields with errors should be cleared on alert and not all fields. **In addition,** ensure that there is a Reset button on the form and when clicked, the Reset button clears the form contents.

- **Ajax, JSON - Use Ajax and JSON to implement a portion of the Student Survey Form to provide a better user experience in filling out the survey form. More specifically, when a user enters a zipcode, populate the city and state fields automatically based on the zip entered, using Ajax. The following is the detailed specification of this requirement.**
  - Please add entries on the Student Survey Form to capture a student's personal information that will allow the department to follow up with students. The personal information will include: Name, Address, Telephone Number, and Email (I think some of this information is already there on the Survey Form). To capture the Address, provide text fields to enter Street and Zipcode and provide place holders for City and State (e.g., use div or span for the City and State, but not text fields). When a user is filling out the form and as soon as he/she is finished typing the Zipcode, populate the corresponding City and State if the entered Zipcode was a valid one. For this assignment, you will need to get the list of valid Zipcodes and corresponding City and States from the server as a JSON file, as a result of an Ajax call. Compare the entered Zipcode value against the data (i.e., Zipcodes) returned from the server. If a match is found, then populate the City and State fields automatically by retrieving the corresponding city and state from the JSON file. Else, display "an invalid zip" message next to the zipcode field.
  - To implement this assignment, you will need to develop a JavaScript function that could be used as an event handler. Register this event handler to handle the "onblur" event associated with the Zipcode element on the Survey Form. The event handler makes an Ajax call to the server to get a list of valid Zipcodes along with corresponding City and State. The Ajax calls returns a JSON representation of an array of objects where each object has the attributes zip, city, state. An example of a JSON object is {"zip": "20148", "city": "Ashburn", "state"= "VA"}. When the Ajax call has completed successfully, invoke a callback function that parses the returned JSON file using either JavaScript's eval function or the JSON parser. Both can be used to convert the returned JSON file into a JavaScript object and then retrieve city and state values for a zipcode. The JSON files have .json extension and is stored on the server in the same folder as your application (e.g., HTML files). (Please refer to sections of Deitel book for further discussion on using JSON. You can also visit Deitel's json resource center at www.deitel.com/json).

    The following is a **sample** JSON file that could be used to implement the homework. You will need to save this information in a file with .json extension and store it on the server.

    ```
    {
    ```

```
    "zipcodes":[
          {
                "zip": "22312",
                "city": "Alexandria",
                "state":"VA"
          },
          {
                "zip": "22030",
                "city": "Fairfax",
                "state":"VA"
          },
          {
                "zip": "22301",
                "city": "Rockville",
                "state":"MD"
          },
          {
                "zip": "20148",
                "city": "Ashburn",
                "state":"VA"
          }
    ]
}
```

Submission

- Please create a link to this homework on your class web site created as part of homework assignment 1. In addition, please submit homework on the Blackboard by providing a zip file with URL and all related files.

- NOTE: We will access each assignment shortly after the due date and not look at your web site afterwards. If you submit an assignment late, you must send an email me and the TA telling us that it is ready (late penalty will apply). You must include "swe642" in the email subject.

- Making your work available to me and the TA is your responsibility; if we cannot access your file then you will not get credit. Be sure to test access to your homework before the due date.

- **Make sure your last name is on *every* HTML page and programming artifacts so we know who it belongs to.**

Grading Rubric

The following areas will be used in the basic grading of the projects:

- Does the solution meet the functional requirements: 80 Points.
- Does the assignment run without errors: 15 points.
- Do the submitted files include comments that include your name as the author: 5 Points.