COMP 2406 B - Fall 2022
Tutorial #2

Due Sunday, October 2, 23:59

Objectives

- Practice basic JavaScript programming using strings, objects, JSON, and functions
- Practice basic HTML and CSS
- Practice creating DOM event handlers

Expectations

You need to submit solutions to all the problems in this tutorial. Our TAs will mark your submission. Remember to use the available resources (w3schools, Node.js documentation, Eloquent JavaScript book, lecture materials, etc.) for more information if you are struggling to complete the problems.

Marking scheme. For each tutorial, you will receive:

- 2/2 for submitting high-quality solutions to all problems. “High-quality” means that your code works (solves the problem) and is also neat and concise (no overengineering, please).
- 1/2 for submitting solutions for all problems but some need improvement, or you are missing a problem.
- 0/2 if you are missing several problems or your solutions are poorly done; or you do not make a submission, or your submission cannot be executed.

Problem 1 (Strong Password)

When creating an account online, most websites require the user to enter a "strong" password. A strong password is a password that someone is unlikely to guess. Strong passwords must be sufficiently long, contain upper and lowercase letters, contain punctuation, etc.

Download the **strongPass.js** file from Brightspace and complete the testPassword() function, which tests the strength of the given password. testPassword() should verify the password meets the criteria below in the order specified. If the criteria is not met, testPassword() should return an appropriate message

indicating what is wrong with the password. If all the criteria are met, testPassword() should return an empty string.

- Minimum length of 6 characters - Use the length property to ensure the password is long enough.
- No spaces - Use indexOf() to ensure the password does not contain any spaces.
- Use at least one digit - Create a loop to examine each character of the password and count how many times a digit character appears. JavaScript does not have a function to verify if a character is a digit, so use the isSingleDigit() function provided. The password should have at least one digit.
- First 3 characters must not be repeated at the end - Use substr() to extract the string at the front and end of the password. Then, compare the substrings with ===.
  Example: Password "abc123abc" is not acceptable because "abc" at the front of the password is the same as "abc" at the end of the password.

The code is currently testing the password "pass" which should fail because the password is only 4 characters long. Verify that testPassword() works by trying passwords that fail each of the four criteria. Keep the passwords, you tested your solution on, as commented out strings at the top of your file for submission.

Problem 2 (Validation)

Download the **validator.js** file from Brightspace. The validCredentials() function contains two parallel arrays of usernames and passwords. Modify validCredentials() to use the indexOf() method to search the usernames array for the given enteredUsername. If the username is found, the same location in the passwords array should contain the enteredPassword. Return true if the passwords are equal, false otherwise. validCredentials() should also return false if the given username was not found.

Problem 3 (Higher Order Array Functions)

Download the **students.js** file from Brightspace. This file contains an array called students, which contains several JavaScript objects representing student information. Add code to this file which uses the higher order array functions (filter(), map(), reduce()) to generate and output the information required below:

1. Generate an array containing the first names of all students. Print it to the console to verify.
2. Generate and print an array containing the full names of students (first and last name separated by " ") of all students who received an exam grade of 80 or higher.
   The students should be `[ 'James Johnson', 'Stephanie Ottesen', 'Leonard Arvan', 'Beverly Mott', 'Beatrice Jaco' ]`
3. Generate and print the total average final grade of all students. Each student's final grade should be calculated with the weighting: assignment=40%, tutorial=10%, exam=50%. The average of all grades should be ~ 71.36%.

If you are struggling, remember to break the problems down into smaller steps (e.g., get the filter() to work first before implementing the map() portion). If you're not sure how to use filter(), map(), or reduce(), check lecture notes or look them up in w3School's JavaScript reference for a refresher:

https://www.w3schools.com/jsref/jsref_obj_array.asp

Problem 4 (Update Rating)

Download the **updateRating.html** file from Brightspace. You can run it in the browser and see what it does. There are 5 blue stars ★ ★ ★ ★ ★ that describe the current rating of the dish. There are also 5 buttons, each updating that rating accordingly.

The code for **updateRating.html** file is an example of terrible programming. Not only do scripts and styles sit inside the HTML file, but also, all the functionality is hardcoded without any reusability of code.

If you decide to change the rating system from 5 stars to 12 you will have a very hard time updating your webpage. Your goal is to rewrite the provided code such that the update from 5 stars to any other number can be done by replacing a value of a **single variable** without making any other changes to the rest of the code. To improve reusability, your JS code should contain exactly one occurrence of the word "blue" (and one of "lightgray").

The body of your HTML file should be:

```
<body onload="init()">
    <h1>Restaurant Dish Review</h1>
    <p id="stars"></p>

    <p><strong>Update rating:</strong>
      <div id="buttons"></div>
    </p>
    <p><strong>Favorite dish:</strong> Ukrainian Borscht</p>

    <img src="ukrainian-borscht.jpg" alt="Ukrainian Borscht" style="width:30%;
float:left">

    <p><strong>Review:</strong> If you don't know what borscht is, it is a deep,
red-coloured soup with cabbage, beets, potatoes, carrots, onion, garlic, and
possibly beef and beans - served with sour cream and dill. Fantastic taste!
Essentially, this beet borscht is a superfood and a meal in itself.
    </p>

    <script src="updateRating.js"></script>
</body>
```

The rest of the HTML elements should be coming from your JavaScript file.

Hint: Below is an example of creating **n** stars using a for-loop and template literals (``):

```
for (let i = 1; i <= n; i++ ){
    result += `<span id="rating${i}">&#9733;</span> `;
}
```

Problem 5 (Save Your Work & Submit)

Keep your files organized. For example, place them into "**Tutorial-02**" folder, where you can easily find them later for your reference and for submission.

To submit your tutorial, you will need to **zip** (compress) all your files for the required tutorial into a single .zip file and submit it to the Tutorial submission on Brightspace. Name your file **T2-YourName.zip**. Make sure you download your .zip file and check its contents after submitting. If your .zip file is missing files or corrupt, you will lose marks.