CS 3744 Spring 2020 - Assignment 3 - Interactivity (200 points)

Due : 3.24.2020 (Tuesday) 11:59PM

Description

In this assignment, you will be given an HTML file with a pre-styled ToDo list. Some tasks have been added/hardcoded for you. The buttons and UI has been designed already, your task is to write a javascript file to add functionality to the buttons and the ToDo List UI. This will require you to understand what elements/containers are on the page, and how to program them correctly for the desired behavior.

The HTML File needs an 'index.js' With that file correctly in place, the ToDo List should perform and have the functionality as shown in the video attached/provided in this document.

Note that the index.html includes two external libraries (jQuery and Bootstrap). Given this is an external URL, it will function correctly only if you have an online connection. If you would like to have a local copy, feel free to download each file and revise HTML. However, you will be asked to only submit index.js and we will test your javascript file with the given index.html

Video:

- Link (use VT account)
- https://drive.google.com/file/d/1aHQTlNvxzR-YuHq6pa26U87zfRds38aq/view?usp=sharing
- https://drive.google.com/open?id=1aXOiOtRUeyrk3XyZdM4cyN8RwmeaKJul

Functionality Expected:

1. Rendering Tasks from given data. (40 pts)
    a. In the given javascript file, you will have an array of tasks. Imagine this is data fetched from a server and your job is rendering the tasks on the table when the document is ready or if there is any change in any states of tasks array. Note that the index.js is included before HTML tags are declared.
    b. Render the whole task data whenever there is any change in any state of tasks array. This means that the program should remove existing todo items before rendering.
    c. Note that you are asked not to revise the HTML file but the program should dynamically add/remove HTML elements.
    d. The child elements of <tbody> tags are good examples for you to know what needs to be rendered programmatically. However, these should be removed as

part of the rendering function. All the rendered tasks should only come from the tasks array given in js code.

e. It should not render any deleted tasks.
f. If a task is overdue, it should use a contextual class "danger" to make the row appear in red.
g. If the title is longer than 30 letters, it should be truncated to the first 30 letters and ellipsis (three dots → ...) Javascript String methods will be useful.
h. Wherever you render tasks, you should maintain the state (completed)
i. Even after you re-render each row, it should have the same behaviors.
j. I recommend using id or value properties for HTML Elements to leave the index of the associated task. This will be useful later to complete tasks so that each element can know how to access the task in the tasks array.
k. I recommend creating a function to render HTML of one task (one row + one row for collapsible note), the argument of which takes one task object. And have a function that iterates over each task in the tasks array.
l. Formatting dates can be tricky. Check out methods of Date objects. Note that the month starts from 0 (January), not 1 (February).

2. Expandable/collapsable note. (10 pts)
    a. Each item has a collapsible/expandable detailed note to show full title and note. Clicking the Note (in yellow) button should toggle the extra row in the table.
    b. It uses Bootstrap Collapsible. As long as the data-target is correctly specified, it should work without any javascript code at all. Follow the format given in the index.html.
3. Completing a task (20 pts)
    a. Checking the checkbox in front of each item indicates that one completed the task.
    b. If a task is completed, the task name should be in strikethrough. Use Bootstrap tag called <del>. Do not use add any style attributes. Also the row should use a contextual class "success" to make the row appear in green
    c. When a task is completed, the completed date of the task should be set with the current time. Check Date object.
    d. If you uncheck the item, it goes back to the incomplete state where completed = false and completeDate is null.
4. Deleting a task (20 pts)
    a. The program should confirm if they are sure.
    b. The trash can icon on the right should delete the selected to-do item regardless of its completed state.
    c. In reality, do not remove it from the data structure, simply mark the deleted flag true so the program uses the flag to skip rendering.
5. Emailing a task (10 pts)
    a. Using the Email button, one should be able to send an email with a title (task title) and the body of the email (task note).
    b. Make sure that each email content will vary per todo-item.

    c. You do not have to worry about line breaks not showing in the email body.
6. Adding a new task (40 pts)
    a. The Add Task button on the bottom right should open the Bootstrap modal dialog. To open (or hide) the modal dialog using javascript/jQuery, please see this.
    b. Once the modal dialog is open, there should be two text input boxes and one text area.
    c. Task Title is required. If no content is typed, the program should alert a user to enter it. Do not create a task in this case and leave the modal input form open.
    d. Due Date should be a proper date format that Date object can parse. Note that it will return NaN if the date cannot be parsed. You may use this as an indicator to see if the user entered the due date in the correct format. The function of `isNaN` should be useful. Here's what should happen when the date format is wrong (link).
    e. Note that the value entered in the textarea can have line breaks (`\r\n`). However, note that HTML will not be able to parse this. Therefore you should replace these line breaks with
 tag when you render it. This StackOverflow page should be useful.
    f. Use createDate property to store the timestamp.
    g. If you cancel it, there should be no change in the data and the modal window should be closed.
    h. You should create an element in the tasks array with a unique index (array length should be useful.)
    i. The tasks should be rendered in the order of elements in the array.
    j. In case you use the submit event of the form (you might not) make sure not to refresh the page so as not to lose any update. (e.g., event.preventDefault()). If your task item does not appear in the list (or appear and disappear) it is likely the problem of the default behavior of <form> refreshing the page when a submit button (inside the form) is pressed. (**3/19/2020 Sang added**) This post will help.
7. Overdue (15 pts)
    a. Using the Overdue button, a user can see only the tasks that are overdue. (Due date is past)
    b. Use the "active" class to indicate the current state (on/off) of the Overdue button. Check examples here.
8. Hide Completed Tasks (15 pts)
    a. Using Hide Completed Task button, a user can filter out completed tasks
    b. Use the "active" class to indicate the current state (on/off) of the Hide Completed Tasks button. Check examples here.
9. Delete Completed Tasks (15 pts)
    a. A user should be able to delete tasks that are completed using the button.
    b. The button should be disabled if there is no completed task.
    c. It should ask the user to confirm if they really want to delete N task(s). Note that the message should be grammatically correct (singular/plural) depending on the number of completed tasks.
10. Bootstrap (15 pts)

    a. Note that we do not use any style attributes or css syntax in HTML.
    b. You are NOT allowed to use CSS or style attributes in javascript either. Use Bootstrap.
    c. The end result should look exactly the same as the given video example.
    d. Note that it is using Bootstrap Grid so it should be responsive (you do not have to do anything to make it responsive.)
11. Extra Points (10pts)
    a. Note that the current example given has no function to “edit” the existing information.
    b. You will get 10 more points if you implement a feature to update existing tasks. You can use any UI you want.
    c. For this part, you may (or may not) have to revise the HTML file as well. Please submit two additional files **SEPARATELY** in addition to index.js which will cover 1~10.
    d. Name files as follows: index-edit.html, index-edit.js. Edit function in index.js will not be graded.

Notes

- You are going to submit ONLY one JS file so do NOT revise the HTML file. We are going to grade with the HTML file of ours with the JS file that you submit.
- We are going to grade in the most recent Chrome.
- For me, it took approximately 240 lines to complete the work.
- Note that the video is also part of the specification. (like Assignment 2).
- Using jQuery is highly recommended but not required. You can use pure javascript.
- Feel free to add more to-do items to begin with. However, do not remove existing ones.