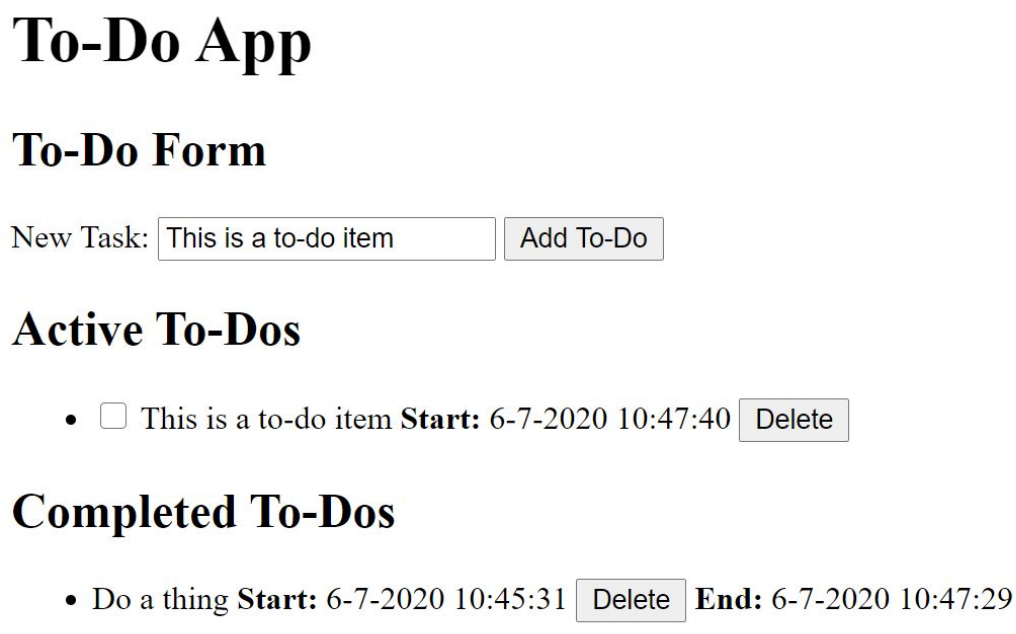
Javascript To-Do List Assignment

Your assignment is to create a To-Do list using HTML and Javascript. CSS is optional. This assignment will focus on DOM manipulation and working with arrays. The following screenshot shows what a To-Do App **might** look like. Your To-Do App can look different. It can have its own style and design and you can add features. This assignment is meant to be flexible to provide you with an opportunity to show your individual talents. Start by building the Mandatory Requirements then expand upon the application as you see fit. Don't forget to make a lot of commits along the way. Even if you are unable to finish we would still like to see your code so don't wait until the end of the project to push your code to github.

Sample Screenshot



Mandatory Requirements:

Please make a "To-Do" JavaScript program that can handle:

- ❏ Adding new to-do items to an “Active To-Dos” list.
- ❏ Completing to-dos by checking a checkbox, or by clicking on the to-do item which then adds the item to a separate "Completed To-Dos" list with the checkbox checked and disabled, and removes the item from the “Active To-Dos” list.
- ❏ The HTML for the page must be semantic.

Challenge Goals:

Challenge goals are optional and are meant for students who would like a more difficult challenge. You are permitted to add any feature you like so long as you have the mandatory requirements completed.

- ❏ Show the To-Do start/add/creation time, and the To-Do end/completion time beside the To-Do. (pictured above)
- ❏ Add a delete button beside each To-Do. (pictured above)
- ❏ Add an edit button beside each To-Do that will allow the text of the To-Do to be edited. If you also support timestamps, add a last edited time. (not pictured)
- ❏ Add a column (before “Active To-Dos”) called “Pending To-Dos” for To-Do items that you haven’t started yet, and add a start button beside each item. Make sure that the start time is only added once the start button is clicked and the item is moved into the “Active To-Dos” column.
- ❏ Don't be afraid to spice it up with some CSS and HTML if you have the chance! (not pictured)

Super Challenge Goal:

- ❏ Turn data into information!
    - ❏ Create another input for Time Estimates that will only accept numbers with decimals rounded to the nearest 0.25 so that you can estimate how long a task is going to take down to the nearest quarter-hour.
    - ❏ When the user completes the task, the app will take the end time, subtract the start time to get the actual time taken for the task. Then the app will compare the actual time vs. the estimated time and display whether the user has spent more or less time than expected, and how much.

Turning data into information is a big part of what learning to program is all about. So what is the difference between data and information? Data is just facts. The time you started a task is a fact, the time you ended a task is also a fact. But when you provide context and meaning for those facts you will gain information. For instance, if we ask the user for a time estimate on each task and compare that estimate with their results we can now declare that a task was completed either ‘early’ or ‘late’. We can track how accurate the user’s estimates are by calculating the difference between their actual time and estimated time. Then we can average out those differences to gain information on how to make more accurate estimates. This assignment’s super-challenge is to turn data into information. I’m looking forward to seeing your results.

Hints

To help you get started on this assignment we're going to describe some code that Warren wrote in the last cohort, which you can see the result of in the screenshot attached to this document. We're only going to describe this code in English and not in pseudo-code and we will not be providing further hints.

- Arrays and corresponding <ul> elements will be needed for each column in your To-Do list
- You will have to create a <form> for collecting the title (and the estimate) for new To-Do items.
- You will have to addEventListener to the form for the submit event.
- Formatting dates into nice human-readable strings is a pain in the butt. You may want an extra function for this. If you make it so the function takes no parameters but instead just finds the current date on its own and then returns the formatted date string then you can use this function for getting both the start and ending times for to-do items.

After the assignment is handed in we will unlock the github repository that stores all of Warren's sample code so that you can see how his solution compares to yours.

Rubric

You will be evaluated on the following points. You must get all 4 of the Mandatory points to pass:

Requirement	**Points**
MANDATORY: ● User can add a to-do item to Active ● User can move a to-do item to Completed ● Timesheet is submitted ● Trello is linked in README.md (Trello must be public for this to count)	4
CHALLENGE: ● Student has added a unique/unanticipated feature ● Student has added timestamps ● Student has added delete feature ● Student has added edit feature, and if timestamps are added, a last edited time is present ● Student has added an additional column and adjusted timestamp behaviour so that start timestamp is added when the item is moved to the Active list, not when the item is created. ● Student has added some nice CSS and HTML	6
SUPER CHALLENGE: ● Student has added field to the form for collecting estimates ● Estimates are rounded to the nearest 0.25 hour ● Estimates are compared to the actual time taken to complete a task and the results are presented and fully explained to the user.	3
Total:	4

Citation Guide

Whenever you borrow code, the following information must be included:

- ❏ Comments to indicate both where the borrowed code begins and ends.
- ❏ A source linking to where you found the code.
- ❏ Your reason for adding the code to your assignment/project instead of writing it out yourself
- ❏ How it works. Explain to us how the code is supposed to work, include links to documentation/articles you read to help you understand.
- ❏ A small demonstration to prove you understand how the code works.

```
const inputArr = [5,1,3,4,2];

/*Borrowed code for bubbleSort starts*/
let bubbleSort = (inputArr) => {
    let len = inputArr.length;
    for (let i = 0; i < len; i++) {
        for (let j = 0; j < len; j++) {
            if (inputArr[j] > inputArr[j + 1]) {
                let tmp = inputArr[j];
                inputArr[j] = inputArr[j + 1];
                inputArr[j + 1] = tmp;
            }
        }
    }
    return inputArr;
};

/*Borrowed code from bubbleSort ends*/

//Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
//Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my own
//How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter9/tutorial
//Demonstration of understanding:
//Example array: [3,1,2]
//Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
//Array: [1,3,2]
//Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
//Array: [1,2,3]
//Step 3: Compare 1 and 2. No need to swap.
//Array: [1,2,3]
//Step 4: Compare 2 and 3. No need to swap.
//Array: [1,2,3]
//Function complete.
console.log(bubbleSort(inputArr));
```