1. Structure of To-Do-List

The structure of your program can be described as follows:

1. HTML Structure:

- The HTML file starts with the `<!DOCTYPE html>` declaration, followed by the `<html>` tag.
- Inside the `<html>` tag, there is the `<head>` section, which includes metadata and the title of the page.
- The `<body>` section contains the visible content of the webpage.
- Within the `<body>` section, there is a `<div>` element with the class "container" that wraps the entire content.
- Inside the container, there is an `<h2>` heading with the text "To Do List".
- Next, there is a `<div>` element with the class "manage-width" that contains an input field for the user to enter tasks and a button to save the task.
- Following that, there is another `<div>` element with the class "toDoListItem" which will display the saved tasks.
- The closing tags for the `<div>` elements and other necessary elements are included.
- Lastly, there is a `<script>` tag that references the JavaScript file.

2. CSS Styling:

- The CSS styles for the program are defined in an external file called "style.css", which is linked to the HTML file.
- The CSS file contains styles for various elements, such as the container, heading, input field, button, and list items.
- The styles define properties like width, margin, padding, font size, color, and more to visually design the elements.

3. JavaScript Logic:

- The JavaScript code is included in a separate file called "index.js", which is referenced by the HTML file.
- The JavaScript code starts by declaring variables `i` and `arr`, which are used for task indexing and storing tasks, respectively.
- The `saveTask()` function is defined to handle the saving of tasks.
- The function checks if the user entered an empty task or if the task already exists in the array. If not, it adds the task to the list and updates the array.
- The `done(el)` function is defined to mark a task as completed by adding a strikethrough style to the task element.
- The `notReq(el)` function removes a task from the list when the user clicks the " ❌button.

That's the overall structure of my program, consisting of HTML markup, CSS styles, and JavaScript logic, working together to create a dynamic to-do list application.

The purpose of a to-do list is to help individuals or groups organize and manage their tasks and responsibilities effectively. Here are some common purposes of using a to-do list:

- Task Management
- Time Management
- Goal Setting and Tracking
- Collaboration and Communication
- Stress Reduction

2. Data Structure / Mechanism used to store and manipulates

In my code i have am using an array (`arr`) to store and manipulate task data. Here's how the array is utilized in my code:

1. Declaration: You initialize an empty array using `let arr = []`.

2. Saving Tasks: When a task is entered by the user and deemed valid, you push the task value (`user_task_value`) into the array using `arr.push(user_task_value)`. This adds the task to the end of the array.
3. Checking for Existing Tasks: To prevent duplicate tasks, you use `arr.includes(user_task_value)` to check if the entered task already exists in the array.
4. Displaying Tasks: Each task is appended to the inner HTML of an element with the ID `saveTask`.
5. Completed Tasks: completed tasks are marked using the `done()` function, which modifies the style of the task element. The array itself is not directly used for tracking completed tasks.

Overall, my code utilizes an array to store and manipulate task data. It allows you to add tasks, check for duplicates, and dynamically display the saved tasks.

3. To implement the functionality to mark a task as completed and update the UI accordingly, I have used follow these steps:

4. **HTML Structure:** Make sure each task element in the HTML has a unique identifier, such as an `id` or a specific class for easier targeting and manipulation.

5. **JavaScript Logic:**
   - Add an event listener to the task elements (e.g., the `<li>` elements) or their associated buttons (e.g., the checkmark button) to trigger the completion functionality.

- In the event listener function, we can access the specific task element that was clicked using **`event.target`** or by passing the task's unique identifier to the function.

- Inside the event listener function, modify the task element's style or add a specific class to visually indicate that it is completed. For example, you can apply a strikethrough style to the text or change the background color.

- Additionally, we may want to update the task's data in your underlying data structure (in this case, the task_arr array) to reflect its completion status.

- Finally, we call a function to update the UI to reflect any changes made. This function can re-render the task list based on the updated data structure

4. To delete a task from the list and ensure that the UI reflects the changes, you can follow these steps:

1. **HTML Structure**: Ensure that each task element in the HTML has a unique identifier, such as an **`id`** or a specific class, to easily target and remove the task from the UI.
2. **JavaScript Logic:**
   - Add an event listener to the delete button associated with each task element to trigger the deletion functionality.

   - In the event listener function, access the specific task element that needs to be deleted using **`event.target`** or by passing the task's unique identifier to the function.

   - Inside the event listener function, remove the task element from the DOM using the **`remove()`** method or by manipulating its parent element.

   - Additionally, remove the task from your underlying data structure (in this case, the **`arr`** task_arr array) to keep it in sync with the UI.

- Finally, call a function to update the UI to reflect the changes. This function should re-render the task list based on the updated data structure.

5. Code

Html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>To-Do-List</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <div class="container">
        <h2>To Do List </h2>
        <div class="manage-width">
            <input type="text" placeholder="Enter Your Task" id="userTask">
            <button onclick="saveTask()" id="saveButton">+</button>
        </div>
        <div class="toDoListItem">
            <p id="saveTask"></p>
        </div>
        </div>
    <script type="text/javascript" src="index.js"></script>
</body>
</html>
```

Style.css

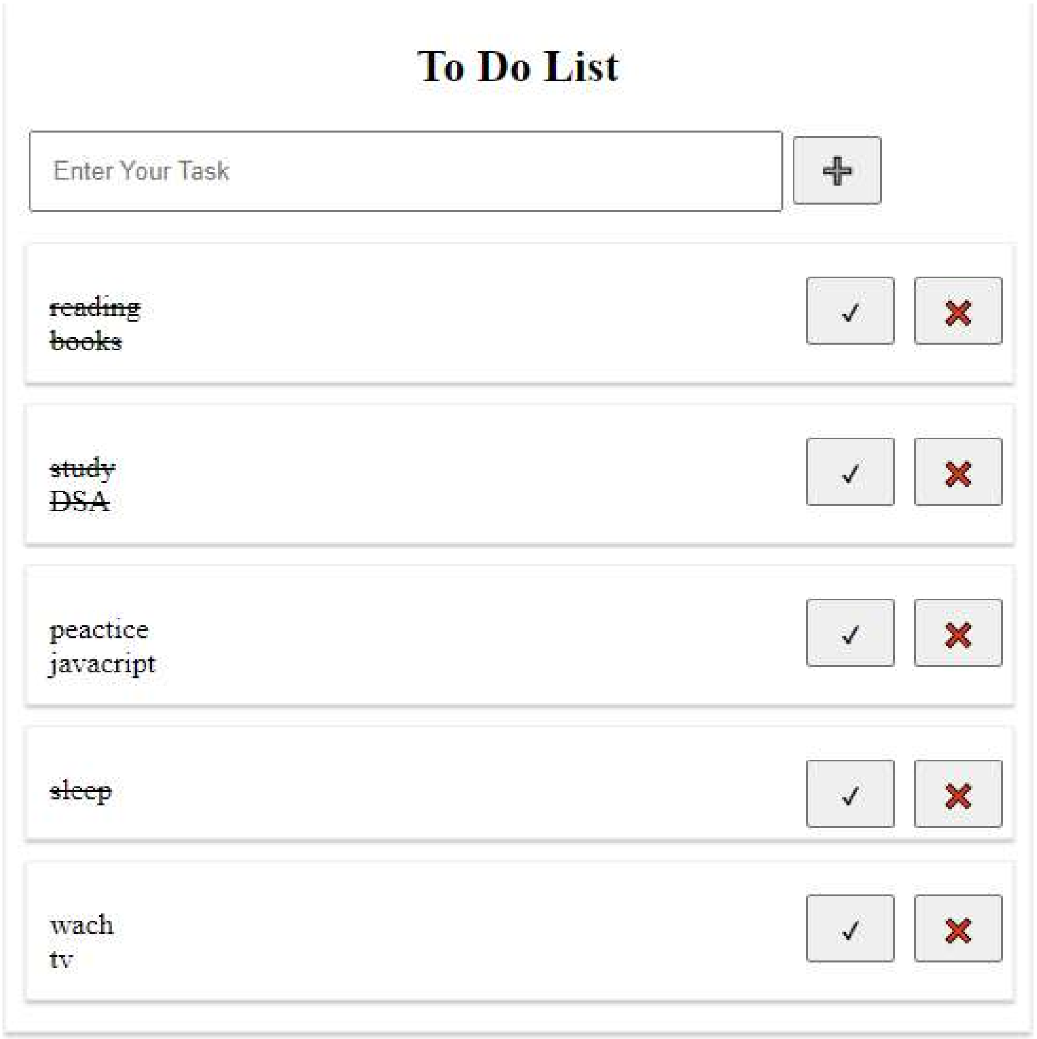
```css
.container{
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 40%;
    border: 1px solid #eee;
    box-shadow: 0 2px 2px #ccc;
}
#userTask{
    padding: 2%;
    width: 69%;
    height: 18px;
    margin-left: 12px;
}
#saveButton{
margin: 1px;
}
.element{
    display: flex;
/*  border: 1px solid blue;*/
    width: 100%;
    padding-top: 12px;
    border: 1px solid #eee;
    box-shadow: 0 2px 2px #ccc;
    margin: 11px 0 11px 0
}
.align-right{
/*  border: 1px solid purple;*/
    float: right;
    width: 100%;
    display: flex;
    justify-content: flex-end;
}
```

```css
li{
    list-style: none;
    padding: 12px;
}
.toDoListItem{
    margin: 10px;
}

button{
    margin: 5px;
    width: 47px;
    height: 36px;}


    .manage-width{
        width: 100%;
    }

    h2{
        display: flex;
        justify-content: center;
    }
```

JavaScript

```javascript
let i=0;
let task_arr =[]
function saveTask() {
    let user_task = document.querySelector("#userTask");
    let user_task_value = user_task.value;
    if(user_task_value===""){
        alert("Empty List cannot be stored");
    }
    else if(task_arr.includes(user_task_value)){
        alert("Task Already Exist")
        user_task.value = ""
    }
    else{
        let save_task = document.querySelector("#saveTask");
        i+=1
        save_task.innerHTML+=`
            <div class="element delete${i}">
            <li id=task${i}>${user_task_value}</li>
            <div class="align-right">
            <button onclick=done(task${i}) id=${user_task_value}" >✔</button>
            <button onclick=notReq('delete${i}')>✖/button>
            </div>
            </div>
        `;
        user_task.value = ""
        task_arr.push(user_task_value)
        console.log(task_arr)
        }
    }
function done(el){
    el.style.textDecoration = "line-through"
}
function notReq(el){
    let d = document.querySelector(`.${el}`)
    // console.log(e)
    d.remove();
}
```

Output



Here is the link : Click Here

https://6470eb3da5aeb30408d71e45--idyllic-speculoos-71ba14.netlify.app/

I have deployed on Netlify platform