**To-do List app**

* First of all we have to define it’s UI

Example –

* Task1
* Task2
* Task3
* Task4

In the text box here user type the task and when it press enter the data will be shown in the tasks list given below the text box. There will also a checkbox when we check it on a task, then it marks that task to be completed. And also a separate delete button for every task, when we delete a task it will be removed from the list.

* Now, we should define the functionalities of the app.

i.e. \* Adding a Task

\* Removing a Task.

\* Check Task

\* Total items count

* After defining functions we can see the data part that where and how the data will be stored for the app.

i.e. Tasks – it will be an array of objects of various tasks and each task object will contain {id, text, done}.

* Defining modules- for performing every different tasks in app we will create different modules or functions for each functionality of app;

I.e. Different modules be like-

- addTodo();

- deleteTodo();

- markAsDone();

- showNotification();

- renderList();

**Let’s start step by step with coding part**

**HTML CODE -**

<html>

<head>

  <title>Todo</title>

  <link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

  <link rel="stylesheet" href="styles.css">

</head>

<body>

  <h1>Todo List App</h1>

  <div id="container">

    <input placeholder="Add a task" class="add-task" id="add" data-helloWord="asdasdas" />

    <span id="total-tasks">Total tasks: <span id="tasks-counter">0</span></span>

    <ul id="list">

      <!-- <li>

          <input type="checkbox" id="task1" data-id="12" class="custom-checkbox">

          <label for="task1">Buy groceries</label>

          <img src="bin.svg" class="delete" data-id="12" />

        </li> -->

    </ul>

  </div>

  <script src="todo.js"></script>

</body>

</html>

**CSS CODE-**

\* {

  box-sizing: border-box;

  font-family: 'Poppins', sans-serif;

}

#container {

  width: 500px;

  margin: 0 auto;

}

ul {

  list-style-type: none;

  margin: 0;

  padding: 0;

  /\* border: 1px solid #e5e5e5; \*/

}

ul li {

  padding: 10px;

  /\* padding-left: 0; \*/

  display: flex;

  justify-content: space-between;

  align-items: inherit;

  align-items: center;

}

ul li:hover {

  background: #f3f3f3;

}

#list input {

  padding: 0;

  height: initial;

  width: initial;

  margin-bottom: 0;

  display: none;

  cursor: pointer;

}

#list label {

  position: relative;

  cursor: pointer;

}

#list label:before {

  content:'';

  -webkit-appearance: none;

  background-color: transparent;

  border: 2px solid #0079bf;

  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.05), inset 0px -15px 10px -12px rgba(0, 0, 0, 0.05);

  padding: 10px;

  display: inline-block;

  position: relative;

  vertical-align: middle;

  cursor: pointer;

  margin-right: 10px;

}

#list input:checked + label:after {

  content: '';

  display: block;

  position: absolute;

  top: 2px;

  left: 9px;

  width: 6px;

  height: 14px;

  border: solid #fff;

  border-width: 0 2px 2px 0;

  transform: rotate(45deg);

}

#list input:checked + label:before {

  background: #0079bf;

}

#list input:checked + label {

  text-decoration: line-through;

}

.delete {

  height: 28px;

    padding: 5px 0px;

    border-radius: 3px;

}

.delete:hover {

  background: #f2f2f2;

  cursor: pointer;

}

h1 {

  text-align: center;

}

.add-task {

  width: 100%;

    outline: none;

    font-size: 15px;

    padding: 11px;

    border: 1px solid #f3f3f3;

    margin-bottom: 3px;

}

#total-tasks {

  margin-bottom: 20px;

  font-size: 13px;

    margin-left: 2px;

    color: gray;

}

**Let’s Start JS Code –**

* **First of all we have to create some variables**

let tasks = []; //this empty task array will store all the tasks data in this array

const tasksList = document.getElementById('list');

const addTaskInput = document.getElementById('add');

const tasksCounter = document.getElementById('tasks-counter');

Variable tasks is created as an empty array to store all the tasks data in it.

tasks[] = [task……….];

and every task is an object

i.e. const task = { text: text; id: date.now().toString(); done: false}

text – it will be the content of task, id- it will make every task unique to perform any action, done- it is set as false because when we have to mark a task as completed it will be set to true;

* **Define different modules for different functionalities of app**

function renderList(task) –this function will be used to render every function on the UI

function toggleTask(taskId) – This function will be used to mark any task to be done

function deleteTask(taskId) – This function will be used to delete any task from the task list

function addTask(task) – This function will be used to add any new task to the list

function showNotification(text) – It will show any notification which will be defined by any function

function handleInputKeypress(e) – it will handle the input from keyboard when a user enters any task.

* First of all we will handle the input because our work start from here when someone start typing in the text box and now we have to start adding tasks which the user has entered
* **keyup** is an event in JavaScript that is triggered when a key on the keyboard is released after being pressed down. In the given code, **someFunction** will be executed when the user releases a key on the keyboard while typing in the **someId** input field.
* The **addEventListener** method is used to register an event listener function to a DOM element, in this case the input field with the ID **someId**. When the specified event, **keyup** in this case, is triggered on the element, the registered event listener function, **someFunction** in this case, is executed.
* So, when a user types into the **someId** input field and releases a key, the **someFunction** function will be executed, allowing you to perform some actions, such as updating the UI or validating the user input.

addTaskInput.addEventListener('keyup', handleInputKeypress);

On line-12 in html file (id- add) is there in the input tag and in js code on line-5 that id is taken by ***addTaskInput*** to perform any input task by js function.

In the above code we are adding an event listener to the ***addTaskInput*** because when a user input something by pressing keys on keyboard it catches that which is the key being pressed by the user and then it will send that event to the ***handleInputKeypress*** function.

* After this line when the event is sent to the handleInputKeypress we will work in this function to retrieve the input data and send this data to add function to add tasks.

function handleInputKeypress(e){

    if(e.key==='Enter'){

        const text = e.target.value;

        if(!text){

            showNotification('Tasks text cannot be empty');

            return;

        }

        const task = {

            text: text,

            id: Date.now().toString(),

            done: false

        }

        e.target.value ="";

        addTask(task);

    }

 }

This function receive the event (e) from the ‘keyup’ event,

Now ***e*** is storing the key input values that which is the key pressed by the user.

We will check that if the key pressed by user is ***‘Enter’*** key then it will store the target value of ***e*** to the variable ***‘text’.***

And also we check that if ***‘text’*** is false then we will show notification that it can’t be empty.

As in the above document we discussed about *tasks array* that it will store ***task object,*** So, Now we will create a task object and in this object it will contain ***text*** that we derived from the keyup event and we will create an unique id, and a Boolean ***“done: false”*** this is because when we have to mark that task to be done then we will set its value to be true.

Now after creating this ***task*** object we set the target value of ***e*** i.e. input box to empty.

And call the ***addTask*** function by passing the ***task*** object in ***addTask(task);***

* **Add Task-**

Now we have to add task in the list, so first of all we have to put the task object that we created by the values that user typed to the array named tasks that we created at very first and render its value to the UI.

function addTask(task) {

    if(task){

        tasks.push(task);

        renderList();

        showNotification('Task added succesfully');

        return;

    }

        showNotification('Something went wrong');

 }

This ***addTask*** function will receive the ***object- task,***

And now if task contains text then we push into ***tasks array*** and ***render it and show a notification*** that task is added otherwise if it doesn’t work then it gives a notification that something went wrong.

* **Delete Task-**

This function works for deleting a task, we will give it a ***task id*** and it will filter the tasks which doesn’t have that same id, and by this process it will filter out and create a new array in which only those tasks which don’t have same id that we passed is present in that array and now we override new array to old array and we get a new array without that task which is to be deleted.

function deleteTask(taskId) {

    const newTasks = tasks.filter(function(task){

        return (task.id!==taskId);

    });

    tasks = newTasks;

    renderList();

    showNotification('Tasks has been successfully deleted');

 }

Here ***task id*** will be the id of task that we created in task object and it will be matched with ***task.id*** then this function will work.

* **Toggle Task / Mark as complete –**

This function will be used to mark any task to be completed or if any task is marked as completed then we can also mark as pending if we want.

It will take ***task id*** and compare it with the id in the ***tasks array*** and if id matched with given id then it will filter out that single task and put it into the *tasks array*, it can have only **one array** because anyways only one id can be matched. So, now we check that arrays length if it is **greater than 0**, then we create a variable as ***currentTask*** and put the value of ***task[0]*** to the current task.

Now we set the value of ***currentTask.done*** to ***!currentTask.done*** to change its value.

function toggleTask(taskId) {

    const task = tasks.filter(function(task){

        return task.id===taskId;

    });

    if(task.length>0){

        const currentTask = tasks[0];

        currentTask.done = !currentTask.done;

        renderList();

        showNotification("Tasks completed");

        return;

    }

    showNotification("Task not completed");

 }

And after changing its value we will render it and gives notification that Task is completed.

* **Rendering –**

It is the most important part of the code because this function is use to render all the functionalities to the frontend or the UI.

And where we interact with all the functions.

function renderList() {

    tasksList.innerHTML ='';

    for(let i=0; i < tasks.length; i++){

        addTaskToDOM(tasks[i]);

    }

    tasksCounter.innerHTML = tasks.length;

}

Let’s recall the variable used in this function,

***tasksList*** – it is the variable in which get the value of ***list id*** in html page on line-14 –

<ul id="list">

***tasksCounter***- it is the variable in which we get the value of ***task counter*** span where we are collecting the number of tasks present in html page on line- 13 –

<span id="total-tasks">Total tasks: <span id**="tasks-counter">**0</span></span>

So, first of all we set the value of task list empty by setting ***innerHTML*** to empty string,

Then we traverse the tasks array and add every tasks to UI or render every task, it done by creating another function which reflect the ***innerHTML*** data to the UI, we will give value of each element of tasks array to the ***addTaskToDOM*** function to reflect it on the UI.

And after that we give value of ***tasks arrays length*** to the **tasksCounter** so it can increase its value and set it to same value as the length of array in tasks array.

* **Add tasks to DOM-**

This function is a subpart of Render, it is as important as render because this function will finally show the functionalities on the UI.

function addTaskToDOM(task) {

    const li = document.createElement('li');

    li.innerHTML = `<input type="checkbox" id="${task.id}" ${task.done? 'checked' : ' '} class="custom-checkbox">

    <label for="${task.id}">${task.text}</label>

    <img src="bin.svg" class="delete" data-id="${task.id}" />`;

    tasksList.append(li);

}

This function takes task as argument. Remember ***task*** is the **object** created in **handleInputKeypress** to store the tasks entered by user in an object.

Now we have to show all the functionalities to the UI, So, we will work on HTML side mixed with JS code.

So for creating a list in JS as html element we will create a variable –

const li = document.createElement('li');

***“li”*** will store the element created by JS which is a list.

Then in innerHTML of that ***“li”*** element we will create a **input tag type of checkbox**, and it’s id will be task.id and we put the **value of text in the label tag** for showing it on the UI. Also we will add an delete image and pass the same taskId which is used for deleting a task.

After creating this ***“li”*** we append this ***innerHTML*** to the ***tasksList*** to show on the screen.

* **Event delegation-**

Above functions will work properly but some functions like delete and toggle function is still incomplete, they are working well but they are not linked with task id on which they have to perform task.

If we click on checkbox it should toggle the value of text.data and also when we click on delete button it should delete the task from the list but it is not working.

We have to make an event that when we click on those elements id will catch its id and return that id to those functions.

Now we create a event listener to the whole document that when there is click anywhere it will return element that where click event is done.

document.addEventListener('click', handleClickEvent);

This will handle click events.

 function handleClickEvent(e) {

    const target = e.target;

    if(target.className === 'delete'){

        const taskId = target.dataset.id;

        deleteTask(taskId);

    }else if (target.className==='custom-checkbox') {

        const taskId = target.id;

        toggleTask(taskId);

        return;

    }

 }

And by ***handleClickEvent()*** we can collect the data of target where click event is done in whole document.

For clean look of code we can collect both event listeners in a function like this -

function initializeApp(){

addTaskInput.addEventListener('keyup', handleInputKeypress); document.addEventListener('click', handleClickEvent);

 }

 initializeApp();