**Documentation Of Task List Project**

This project is about simple web project where user can add some task than delete those task individually also delete entire project list using ***“Clear Task”*** button.

Most important JavaScript code explanation line by line given here for better understanding.

This is the explanation of entire project code here. We do not need HTML code that's why I focus on the JS code.

***Define UI elements:->*** This comment indicates that the following lines of code define variables representing various UI elements in the HTML document.

***\*let form = document.querySelector('#task\_form');:*** This line selects the form element with the ID "task\_form" using the querySelector method and assigns it to the variable form.

***\*let taskList = document.querySelector('ul#task\_li');:*** This line selects the unordered list element with the ID "task\_li" using the querySelector method and assigns it to the variable taskList.

***\*let clearBtn = document.querySelector('#clear\_task\_btn');:*** This line selects the button element with the ID "clear\_task\_btn" using the querySelector method and assigns it to the variable clearBtn.

***\*let filter = document.querySelector('#task\_filter');:*** This line selects the input element with the ID "task\_filter" using the querySelector method and assigns it to the variable filter.

***\*let taskInput = document.querySelector('#new\_task');:*** This line selects the input element with the ID "new\_task" using the querySelector method and assigns it to the variable taskInput.

***Define event listeners:*** This comment indicates that the following lines of code define event listeners for different UI elements.

***\*form.addEventListener('submit', addTask);:*** This line adds an event listener to the form element that listens for the "submit" event and calls the addTask function when the form is submitted.

***\*taskList.addEventListener('click', removeTask);:*** This line adds an event listener to the taskList (ul) element that listens for the "click" event and calls the removeTask function when a click event occurs within the taskList.

***clearBtn.addEventListener('click', clearTask);:*** This line adds an event listener to the clearBtn (button) element that listens for the "click" event and calls the clearTask function when the button is clicked.

***\*filter.addEventListener('keyup', filterTask);:*** This line adds an event listener to the filter (input) element that listens for the "keyup" event (when a key is released) and calls the filterTask function.

***\*document.addEventListener('DOMContentLoaded', getTasks);:*** This line adds an event listener to the document object that listens for the "DOMContentLoaded" event (when the initial HTML document has been completely loaded and parsed) and calls the getTasks function.

***\* Define functions:*** This comment indicates that the following lines of code define several functions used in the application.

***\*Add Task:*** This comment provides a description of the addTask function.

***\*function addTask(e) { ... }:*** This line starts the definition of the addTask function that takes an event object e as a parameter.

***\*e.preventDefault();:*** This line prevents the default form submission behavior, which typically refreshes or redirects the page.

***\*if (taskInput.value === '') { ... }:*** This conditional statement checks if the value of the taskInput is empty.

***\*alert('Add a task!');:*** This line displays an alert message if the task input is empty, prompting the user to add a task.

***\*let li = document.createElement('li');:*** This line creates a new list item element using the createElement method and assigns it to the variable li.

***\*li.appendChild(document.createTextNode(taskInput.value + ' '));:*** This line creates a text node containing the value of the taskInput concatenated with a space and appends it as a child to the li element.

***\*let link = document.createElement('a');:*** This line creates a new anchor element using the createElement method and assigns it to the variable link.

***\*link.setAttribute('href', '#');:*** This line sets the href attribute of the link element to '#' using the setAttribute method.

***\*link.innerHTML = 'x';:*** This line sets the innerHTML property of the link element to 'x', displaying the 'x' symbol as the link text.

***\*li.appendChild(link);:*** This line appends the link element as a child to the li element.

***\*taskList.appendChild(li);:*** This line appends the li element as a child to the taskList element, adding the new task to the task list.

***\*storeTaskInLocalStorage(taskInput.value);:*** This line calls the storeTaskInLocalStorage function, passing the value of the taskInput as an argument to store the task in local storage.

***\*taskInput.value = '';:*** This line clears the value of the taskInput after the task has been added.

***\* Remove Task:*** This comment provides a description of the removeTask function.

***\*function removeTask(e) { ... }:*** This line starts the definition of the removeTask function that takes an event object e as a parameter.

***\*if(e.target.hasAttribute("href")) { ... }:*** This conditional statement checks if the target element that triggered the event has an attribute called "href".

***\*if(confirm("Are you sure?")) { ... }:*** This line displays a confirmation dialog box asking the user to confirm the task removal.

***\*let ele = e.target.parentElement;:*** This line assigns the parent element of the target element (the list item) to the variable ele.

***\*ele.remove();:*** This line removes the ele element (the list item) from the DOM, effectively removing the task from the task list.

***\*removeFromLS(ele);:*** This line calls the removeFromLS function, passing the ele element as an argument to remove the task from local storage.

***\*Clear Task:*** This comment provides a description of the clearTask function.

function clearTask(e) { ... }: This line starts the definition of the clearTask function that takes an event object e as a parameter.

***\*while(taskList.firstChild) { ... }:*** This while loop iterates as long as the taskList element has a first child.

***\*taskList.removeChild(taskList.firstChild);:*** This line removes the first child of the taskList element (i.e., the list items) iteratively until all list items are removed.

***\*localStorage.clear();:*** This line clears the entire local storage, removing all tasks stored in it.

***\*Filter Function:*** This comment provides a description of the filterTask function.

***\*function filterTask(e) { ... }:*** This line starts the definition of the filterTask function that takes an event object e as a parameter.

***\*let text = e.target.value.toLowerCase();:*** This line retrieves the value of the input field (e.target) and converts it to lowercase, storing it in the text variable.

***\*document.querySelectorAll("li").forEach(task => { ... });:*** This line selects all list items (li elements) in the document and iterates over them using the forEach method.

***\*let item = task.firstChild.textContent;:*** This line retrieves the text content of the first child of each list item (task) and stores it in the item variable.

***\*if(item.toLowerCase().indexOf(text) != -1) { ... }:*** This conditional statement checks if the item (task) contains the text (filter value) by comparing their lowercase versions using the indexOf method. If the text is found within the item, it sets the task's style display to "block" to show it; otherwise, it sets the display to "none" to hide it.

***\*// Store in Local Storage:*** This comment provides a description of the storeTaskInLocalStorage function.

***\*function storeTaskInLocalStorage(task) { ... }:*** This line starts the definition of the storeTaskInLocalStorage function that takes a task as a parameter.

let tasks;: This line declares a variable tasks to store the tasks from local storage.

***\*if (localStorage.getItem('tasks') === null) { ... }:*** This conditional statement checks if the 'tasks' key in local storage is null, indicating no tasks are stored.

***\*tasks = [];:*** This line assigns an empty array to the tasks variable if no tasks are found in local storage.

***\*else { ... }:*** This block is executed if tasks are found in local storage.

***\*tasks = JSON.parse(localStorage.getItem('tasks'));:*** This line retrieves the tasks from local storage, parses the JSON string representation into an array, and assigns it to the tasks variable.

***\*tasks.push(task);:*** This line adds the new task to the tasks array.

***\*localStorage.setItem('tasks', JSON.stringify(tasks));:*** This line updates the 'tasks' key in local storage by converting the tasks array into a JSON string using JSON.stringify and storing it.

***\*function getTasks() { ... }:*** This line starts the definition of the getTasks function.

***\*let tasks;:*** This line declares a variable tasks to store the tasks from local storage.

***\*if (localStorage.getItem('tasks') === null) { ... }:*** This conditional statement checks if the 'tasks' key in local storage is null, indicating no tasks are stored.

***\*tasks = [];:*** This line assigns an empty array to the tasks variable if no tasks are found in local storage.

***\*else { ... }:*** This block is executed if tasks are found in local storage.

***\*tasks = JSON.parse(localStorage.getItem('tasks'));:*** This line retrieves the tasks from local storage, parses the JSON string representation into an array, and assigns it to the tasks variable.

***\*tasks.forEach(task => { ... });:*** This line iterates over each task in the tasks array using the forEach method.

***\*let li = document.createElement('li');:*** This line creates a new list item element using the createElement method and assigns it to the variable li.

***\*li.appendChild(document.createTextNode(task + ' '));:*** This line creates a text node containing the current task from the iteration concatenated with a space and appends it as a child to the li element.

***\*let link = document.createElement('a');:*** This line creates a new anchor element using the createElement method and assigns it to the variable link.

***\*link.setAttribute('href', '#');:*** This line sets the 'href' attribute of the link element to '#' to create a placeholder link.

***\*link.innerHTML = 'x';:*** This line sets the inner HTML of the link element to 'x', representing a delete symbol.

***\*li.appendChild(link);:*** This line appends the link element as a child to the li element.

***\*taskList.appendChild(li);:*** This line appends the li element (representing a task) as a child to the taskList element, displaying it in the task list.

***\*// Remove from Local Storage:*** This comment provides a description of the removeFromLS function.

***\*function removeFromLS(taskItem) { ... }:*** This line starts the definition of the removeFromLS function that takes a task item as a parameter.

***\*let tasks;:*** This line declares a variable tasks to store the tasks from local storage.

***\*if (localStorage.getItem('tasks') === null) { ... }:*** This conditional statement checks if the 'tasks' key in local storage is null, indicating no tasks are stored.

***\*tasks = [];:*** This line assigns an empty array to the tasks variable if no tasks are found in local storage.

***\*else { ... }:*** This block is executed if tasks are found in local storage.

***\*tasks = JSON.parse(localStorage.getItem('tasks'));:*** This line retrieves the tasks from local storage, parses the JSON string representation into an array, and assigns it to the tasks variable.

***\*let li = taskItem;:*** This line assigns the taskItem parameter (representing the task's list item) to the li variable.

***\*li.removeChild(li.lastChild);:*** This line removes the last child node of the li element, which is the delete link ('x') associated with the task.

***\*tasks.forEach((task, index) => { ... });:*** This line iterates over each task in the tasks array using the forEach method, along with the index of each task.

***\*if (li.textContent.trim() === task) { ... }:*** This conditional statement checks if the text content of the li element (trimmed of whitespace) matches the current task.

***\*tasks.splice(index, 1);:*** This line removes the task from the tasks array at the specified index using the splice method.

***\*localStorage.setItem('tasks', JSON.stringify(tasks));:*** This line updates the 'tasks' key in local storage by converting the tasks array into a JSON string using JSON.stringify and storing it.

The provided JavaScript code handles adding, removing, clearing, and filtering tasks. It also utilizes local storage to persist the tasks even after page reloads.