@codingbloodhound

Local Storage In JavaScript

Muneer Ahmed
@codingbloodhound

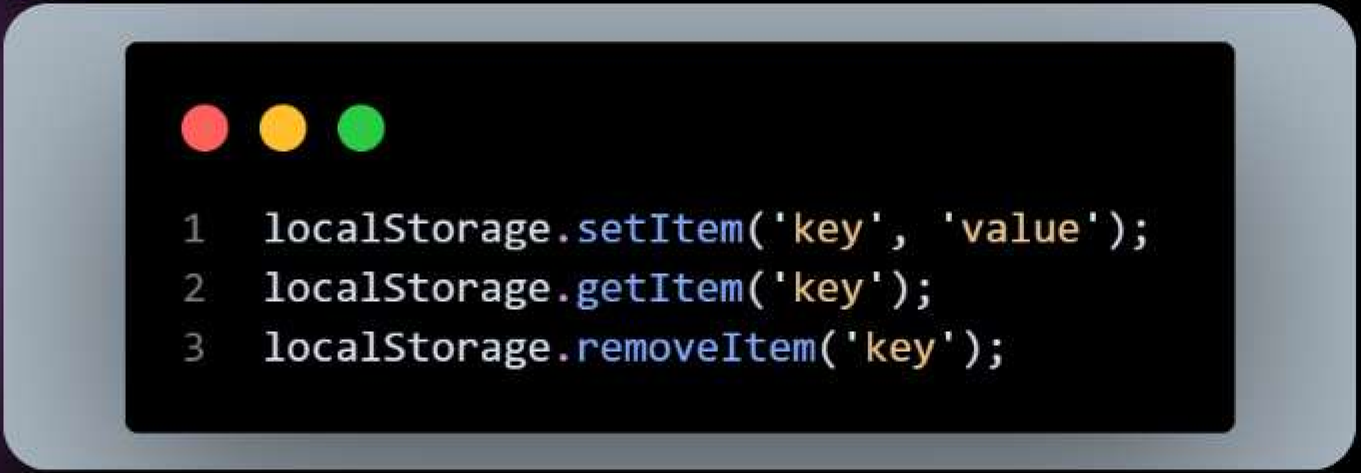
@codingbloodhound

🔍 What is Local Storage?

Local Storage allows you to store data in the browser with no expiration time.

- It’s synchronous.
- Stores strings (you need to JSON.stringify for objects).
- Data remains even after page reloads or closing browser.

Use it with:



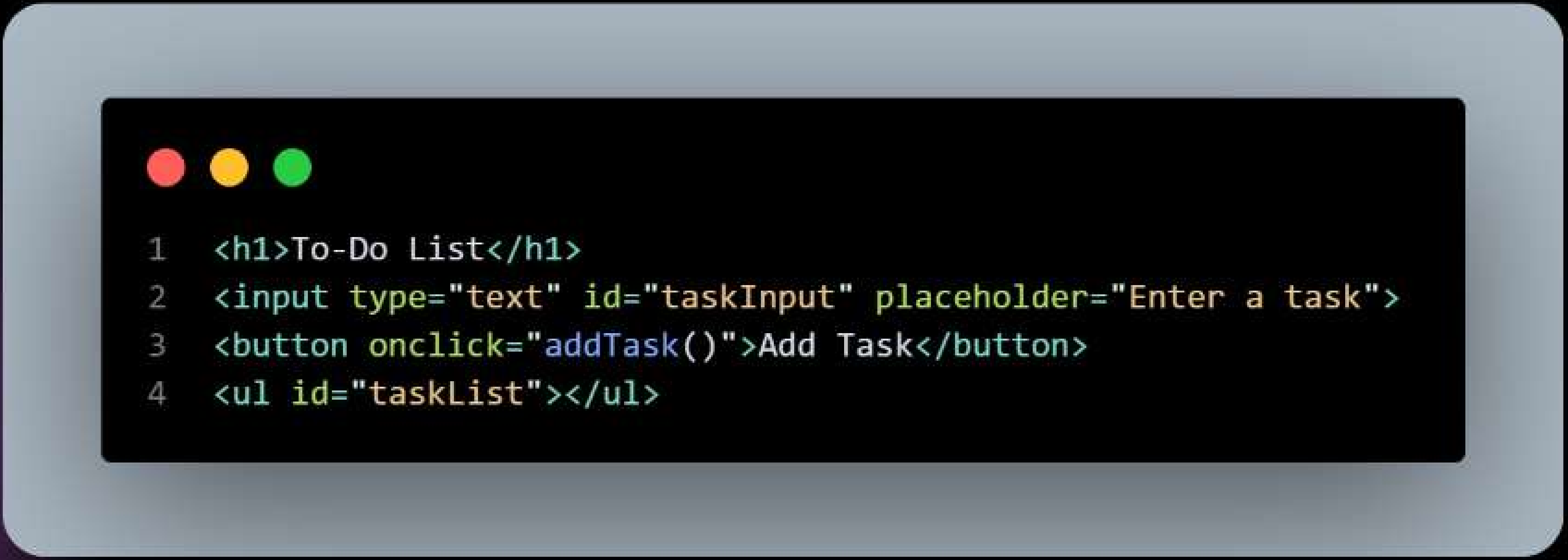
```
localStorage.setItem('key', 'value');
localStorage.getItem('key');
localStorage.removeItem('key');
```

Muneer Ahmed
@codingbloodhound

@codingbloodhound

🎯 Project: To-Do List (with Local Storage)

🔧 Step 1: HTML Setup

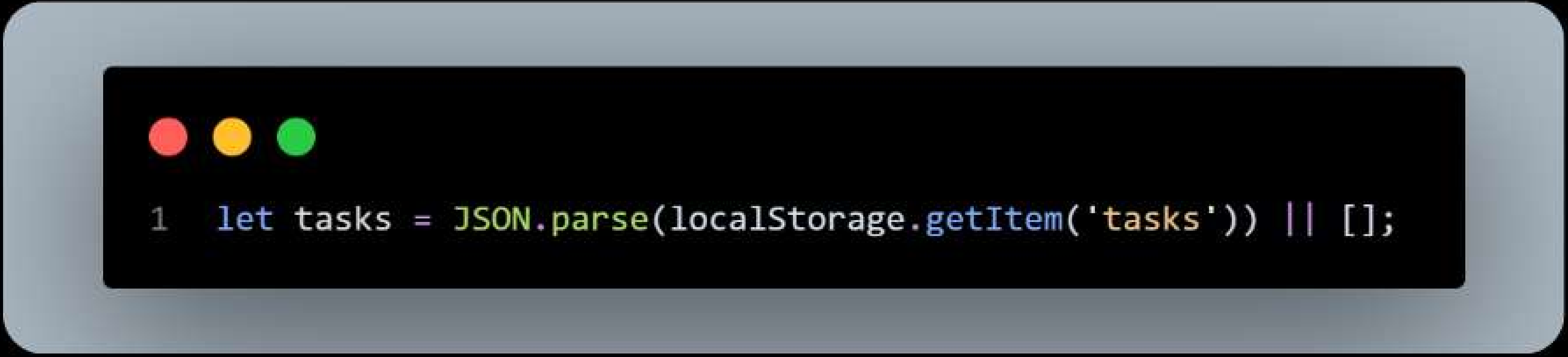


```html
<h1>To-Do List</h1>
<input type="text" id="taskInput" placeholder="Enter a task">
<button onclick="addTask()">Add Task</button>
<ul id="taskList"></ul>
```

Muneer Ahmed
@codingbloodhound

@codingbloodhound

⚙️ Step 2: JavaScript Logic

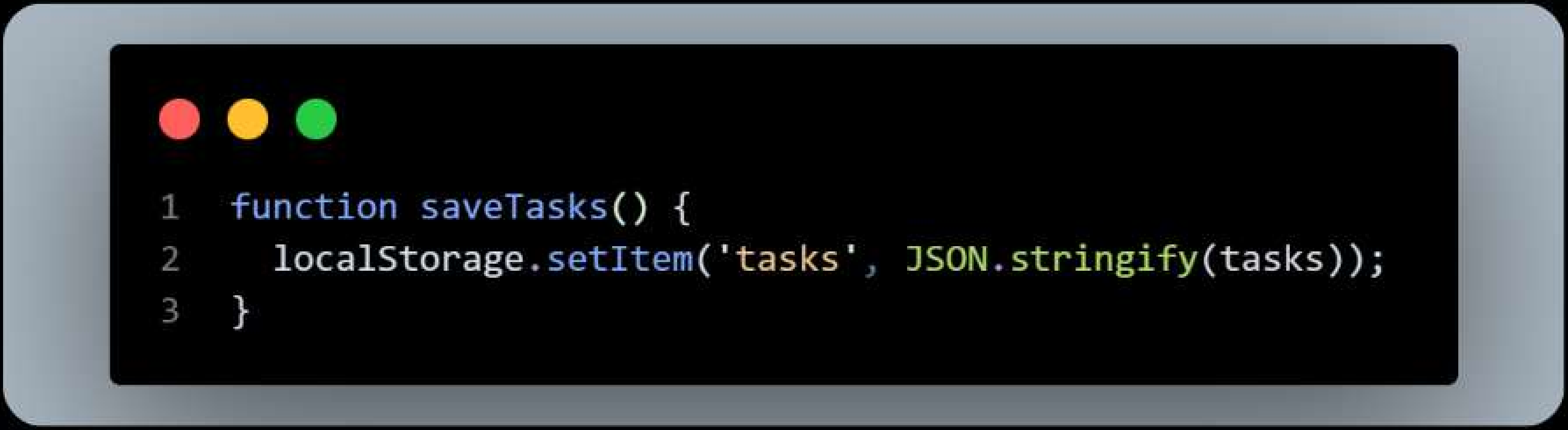


```javascript
let tasks = JSON.parse(localStorage.getItem('tasks')) || [];
```

- Tries to get previously saved tasks from localStorage
- If none found, initializes tasks as an empty array ([])
- Uses JSON.parse() to convert the stored string back to an array

Muneer Ahmed
@codingbloodhound

⚙️ Step 2: JavaScript Logic (save tasks)

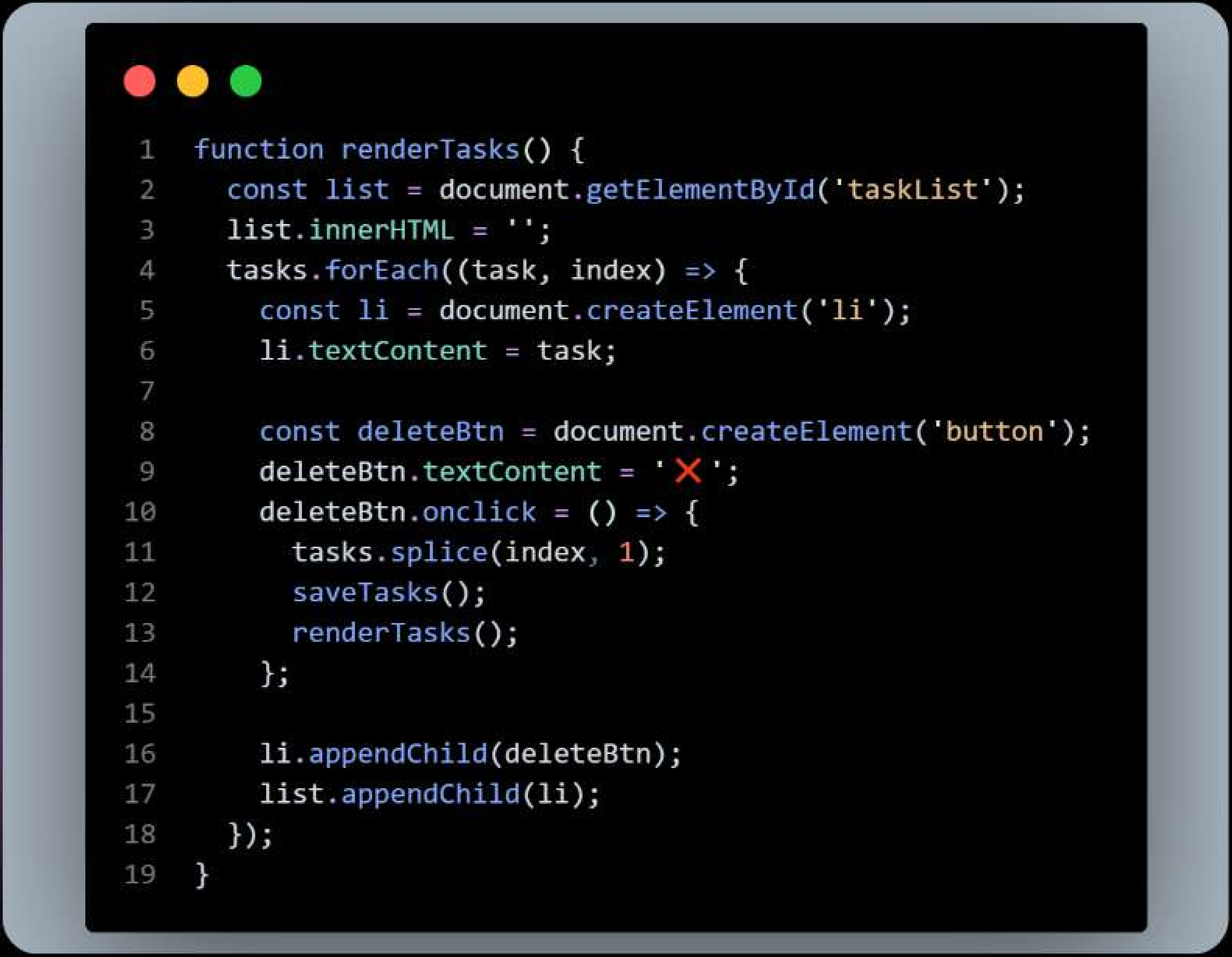

```
function saveTasks() {
  localStorage.setItem('tasks', JSON.stringify(tasks));
}
```

- Converts the tasks array into a string using JSON.stringify()
- Stores it in localStorage with the key 'tasks'

Muneer Ahmed
@codingbloodhound

@codingbloodhound

⚙️ Step 2: JavaScript Logic (render task.)



```
function renderTasks() {
  const list = document.getElementById('taskList');
  list.innerHTML = '';
  tasks.forEach((task, index) => {
    const li = document.createElement('li');
    li.textContent = task;

    const deleteBtn = document.createElement('button');
    deleteBtn.textContent = '❌';
    deleteBtn.onclick = () => {
      tasks.splice(index, 1);
      saveTasks();
      renderTasks();
    };

    li.appendChild(deleteBtn);
    list.appendChild(li);
  });
}
```

Muneer Ahmed
@codingbloodhound

@codingbloodhound

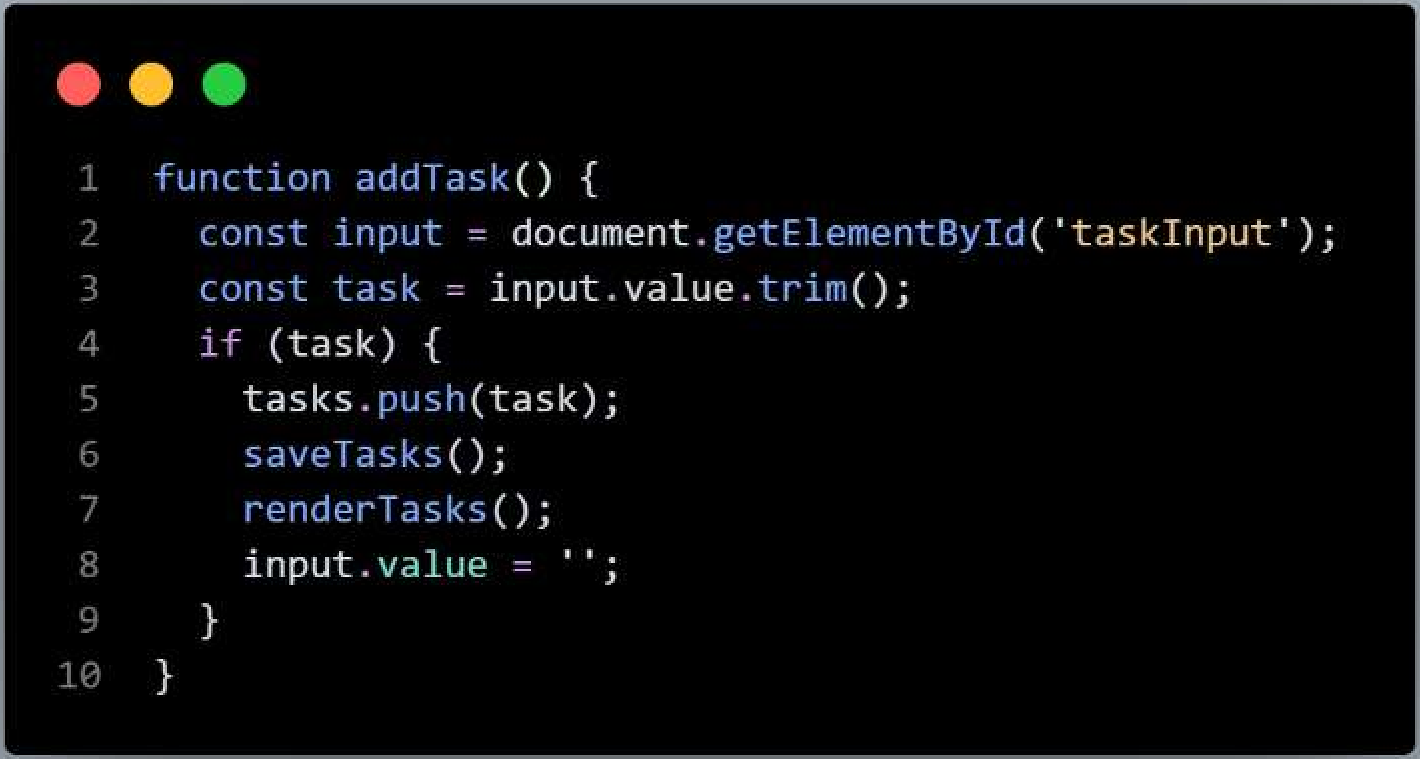
⚙️ Step 2: JavaScript Logic (render task explain code)

- Clears the current task list on the page
- Loops through all tasks in the 'tasks' array
- For each task:
  - Creates a new <li> element
  - Adds task text inside it
  - Creates a ❌ delete button
  - On click of delete button:
    - Removes task from the array using splice()
    - Saves updated array to localStorage
    - Renders updated list again
- Appends each <li> to the <ul> with id taskList

Muneer Ahmed
@codingbloodhound

@codingbloodhound

⚙️ Step 2: JavaScript Logic (Add task)



```javascript
function addTask() {
  const input = document.getElementById('taskInput');
  const task = input.value.trim();
  if (task) {
    tasks.push(task);
    saveTasks();
    renderTasks();
    input.value = '';
  }
}
```

Muneer Ahmed
@codingbloodhound

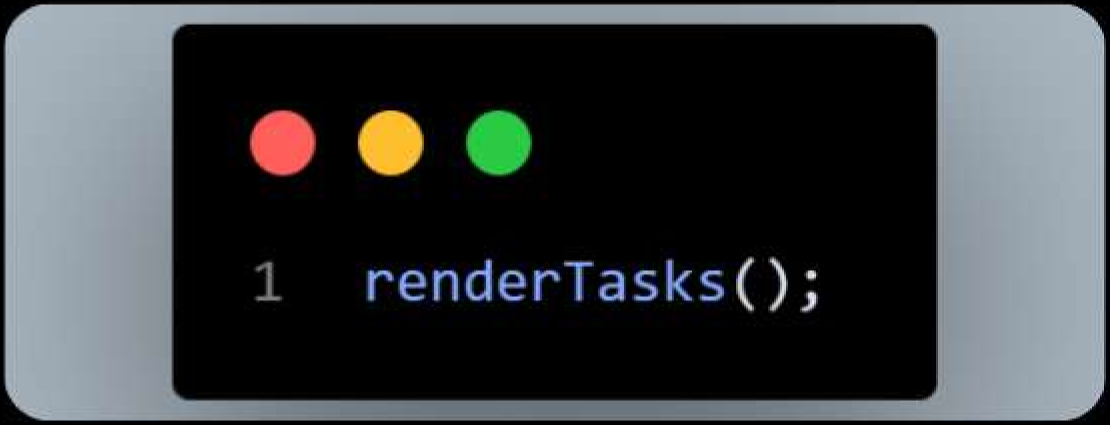
⚙️ Step 2: JavaScript Logic (add task explain code)

- Gets the value from the <input> with *id* taskInput
- Trims any extra spaces
- If input is not empty:
  - Adds the task to the 'tasks' array
  - Saves it to localStorage
  - Renders the new list
  - Clears the input field for new entry

Muneer Ahmed
@codingbloodhound

@codingbloodhound

⚙️ Step 2: JavaScript Logic (Initial Rendering)



```
renderTasks();
```

- Calls renderTasks() once when the script runs
- This makes sure the previously saved tasks (if any) are shown immediately when the page loads

Muneer Ahmed
@codingbloodhound

@codingbloodhound

🚀 What You Learn from This:

- How to use localStorage to persist data
- How to use JSON.stringify() and JSON.parse()
- DOM manipulation with JavaScript
- Building real-world mini projects

💡 Bonus Tips for Learners:

- Try adding an "Edit" button next!
- Add timestamp to each task.
- Make the UI responsive.

Muneer Ahmed
@codingbloodhound

@codingbloodhound

CALL-TO-ACTION

want to Learn More?

Follow me for more tips on Web Development!

Muneer Ahmed

@codingbloodhound