OpenAI

Task Manager App Guide

Let's break it down step by step to make sure you understand the logic behind building a **Task Manager (To-Do List App)**.

📌 Features of the To-Do List App

1. **Add a Task** – Users can enter a task and add it to the list.
2. **Delete a Task** – Users can remove tasks individually.
3. **Mark a Task as Completed** – Users can mark tasks as completed with a visual indicator (e.g., strikethrough text).
4. **Persist Data using localStorage** – Tasks should be saved so that they remain even after refreshing the page.

🔧 Step-by-Step Development

We'll use:

- **HTML** → Structure of the app
- **CSS** → Styling the app
- **JavaScript** → Logic for adding, deleting, marking tasks, and saving them

---

📜 1. HTML Structure

We'll create a simple form to enter tasks and a list to display them.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Task Manager</h1>
        <div class="task-input">
            <input type="text" id="taskInput" placeholder="Enter a task">
            <button id="addTaskBtn">Add Task</button>
        </div>
        <ul id="taskList"></ul>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

🎨 2. CSS Styling

This will give the app a clean and modern look.

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Arial, sans-serif;
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

.container {
    width: 350px;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    text-align: center;
    margin-bottom: 20px;
}

.task-input {
    display: flex;
    gap: 10px;
}

input {
    flex: 1;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

button {
    padding: 8px 12px;
    border: none;
    background: #28a745;
    color: white;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background: #218838;
}

ul {
    list-style: none;
    margin-top: 20px;
}

li {
    display: flex;
    justify-content: space-between;
    background: #e9ecef;
    padding: 8px;
    margin-top: 5px;
    border-radius: 5px;
}

.completed {
    text-decoration: line-through;
    color: gray;
}
```

```css
.delete-btn {
    background: #dc3545;
    color: white;
    border: none;
    padding: 5px;
    cursor: pointer;
    border-radius: 3px;
}

.delete-btn:hover {
    background: #c82333;
}
```

🛠️ 3. JavaScript Logic

Now, let's write the **script.js** file to handle user interactions.

```js
js

document.addEventListener("DOMContentLoaded", loadTasks);

const taskInput = document.getElementById("taskInput");
const addTaskBtn = document.getElementById("addTaskBtn");
const taskList = document.getElementById("taskList");

addTaskBtn.addEventListener("click", addTask);
taskList.addEventListener("click", manageTask);

function addTask() {
    const taskText = taskInput.value.trim();
    if (taskText === "") {
        alert("Please enter a task!");
        return;
    }

    const task = { text: taskText, completed: false };
    const tasks = getTasksFromStorage();
    tasks.push(task);
    saveTasksToStorage(tasks);

    renderTasks();
    taskInput.value = "";
}

function manageTask(event) {
    if (event.target.classList.contains("delete-btn")) {
        deleteTask(event.target.parentElement);
    } else if (event.target.tagName === "LI") {
        toggleComplete(event.target);
    }
}

function deleteTask(taskElement) {
    const tasks = getTasksFromStorage();
    const updatedTasks = tasks.filter(task => task.text !== taskElement.firstChild.textContent);
    saveTasksToStorage(updatedTasks);
    renderTasks();
}

function toggleComplete(taskElement) {
    const tasks = getTasksFromStorage();
    const taskText = taskElement.firstChild.textContent;
    tasks.forEach(task => {
        if (task.text === taskText) {
            task.completed = !task.completed;
        }
    });
    saveTasksToStorage(tasks);
    renderTasks();
}
```

```
function getTasksFromStorage() {
    return JSON.parse(localStorage.getItem("tasks")) || [];
}

function saveTasksToStorage(tasks) {
    localStorage.setItem("tasks", JSON.stringify(tasks));
}

function renderTasks() {
    taskList.innerHTML = "";
    const tasks = getTasksFromStorage();
    tasks.forEach(task => {
        const li = document.createElement("li");
        li.textContent = task.text;
        if (task.completed) {
            li.classList.add("completed");
        }
        const deleteBtn = document.createElement("button");
        deleteBtn.textContent = "X";
        deleteBtn.classList.add("delete-btn");
        li.appendChild(deleteBtn);
        taskList.appendChild(li);
    });
}

function loadTasks() {
    renderTasks();
}
```

🔥 How it Works

1️⃣ Adding a Task

- The user types in a task and clicks **"Add Task"**.
- The task is saved in **localStorage**.
- The UI is updated dynamically to show the new task.

2️⃣ Deleting a Task

- When the **"X"** button is clicked, the task is removed from **localStorage** and from the UI.

3️⃣ Marking a Task as Completed

- Clicking on a task toggles a **completed** class that adds a strikethrough effect.
- The task's completed state is saved in **localStorage**.

4️⃣ Persisting Data with localStorage

- When the page loads, tasks from **localStorage** are fetched and displayed.

🏆 What You Learn

- **DOM Manipulation** (Adding event listeners, modifying elements)
- **localStorage API** (Storing and retrieving data)
- **Event Handling** (Click events for adding, deleting, toggling tasks)
- **Array Methods** (`filter()`, `forEach()`, `push()`)
- **Best Practices** (Separation of concerns with functions, proper data management)

🚀 Final Code (Copy & Paste)

HTML: `index.html`

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Task Manager</h1>
        <div class="task-input">
            <input type="text" id="taskInput" placeholder="Enter a task">
            <button id="addTaskBtn">Add Task</button>
        </div>
        <ul id="taskList"></ul>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

CSS: `styles.css`

(Refer to the CSS section above)

JavaScript: `script.js`

(Refer to the JavaScript section above)

✅ Next Steps

1. Try adding animations using CSS.
2. Add a **"Clear All Tasks"** button.
3. Use **date-time stamps** for tasks.
4. Implement a **drag-and-drop feature** for sorting tasks.

This project covers **core JavaScript concepts** and is a great way to learn! Let me know if you need any modifications. 🚀