A close up of a logo

Description automatically generated

**JavaScript Basics**

**Disclaimer: The content is curated from online/offline resources and used for educational purpose only**

**LAB MANUAL**

**Creating Interactive Web Apps to fetch API data and dynamic To-Do list creation**

**Objective:**

The objective of this activity is to help learners combine fundamental JavaScript concepts — DOM manipulation, event handling, and API fetching — into a single interactive web app. Participants will first fetch and display data from a public API, and then create an interactive to‑do list where users can add, mark, and delete tasks. This reinforces the integration of data handling with dynamic content updates on a webpage.

**Equipment Required:**

* A computer with a text editor (VS Code, Sublime Text, or Notepad++)
* A modern web browser (Chrome, Firefox, Edge, etc.)

**Prerequisites:**

* Basic knowledge of HTML elements and forms
* Understanding of basic JavaScript DOM methods and events
* Familiarity with fetch() for API requests
* Ability to link external JavaScript files to HTML

**Problem Statement:**

You need to build a simple interactive web page that has two main features:

1. A button that, when clicked, fetches and displays data from a public API (e.g., random user info or a random joke).
2. A to‑do list where users can add a task, mark it as completed, and remove it from the list.

This will give learners practical experience in handling asynchronous data fetching and dynamic DOM updates from user input.

**Procedure:**

1. Create two files: ***app.html*** and ***script.js***.
2. In app.html, set up basic structure with a section for API data and another section for the to‑do list UI.
3. In script.js, write a function to use fetch() to get data from an API and display it.
4. Create functions to add tasks, mark them complete, and delete them.
5. Attach event listeners for the "Fetch Data" and "Add Task" buttons.
6. Style the components for a clean, user‑friendly interface.
7. Test in the browser and improve responsiveness as needed.

**Code**

***app.html***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Interactive Web App</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

padding: 0;

margin: 0;

min-height: 100vh;

background: #fff;

}

h1 {

text-align: center;

margin: 0;

padding: 32px 0 18px 0;

color: #333;

font-size: 2.2rem;

letter-spacing: 1px;

}

.container {

display: flex;

justify-content: center;

align-items: flex-start;

min-height: 70vh;

gap: 20px;

flex-wrap: wrap;

}

.box {

background: #eaf6ff;

padding: 32px 28px 28px 28px;

border-radius: 18px;

box-shadow: 0 8px 32px rgba(0, 123, 255, 0.13);

width: 370px;

margin-top: 32px;

transition: box-shadow 0.3s;

}

.box:hover {

box-shadow: 0 12px 40px rgba(0, 123, 255, 0.18);

}

.box h2 {

color: #007bff;

margin-bottom: 18px;

text-align: center;

letter-spacing: 1px;

}

#taskInput {

padding: 12px;

width: 68%;

margin-right: 7px;

border-radius: 7px;

border: 1.5px solid #007bff;

font-size: 16px;

outline: none;

background: #fff;

transition: border 0.2s;

}

#taskInput:focus {

border: 1.5px solid #90caf9;

background: #eaf6ff;

}

#addTaskBtn {

padding: 12px 18px;

border-radius: 7px;

border: none;

background: linear-gradient(90deg, #2196f3 0%, #90caf9 100%);

color: #fff;

font-weight: 600;

font-size: 16px;

cursor: pointer;

box-shadow: 0 2px 6px rgba(33, 150, 243, 0.13);

transition: background 0.2s, transform 0.2s;

}

#addTaskBtn:hover {

background: linear-gradient(90deg, #90caf9 0%, #2196f3 100%);

color: #111;

transform: translateY(-2px) scale(1.04);

}

#taskList {

margin-top: 18px;

padding: 0;

}

#taskList li {

list-style: none;

padding: 12px 14px;

margin: 7px 0;

background: #e3f2fd;

display: flex;

justify-content: space-between;

align-items: center;

border-radius: 7px;

cursor: pointer;

font-size: 16px;

box-shadow: 0 1px 3px rgba(248, 181, 0, 0.08);

transition: background 0.2s;

}

#taskList li:hover {

background: #bbdefb;

}

#taskList li.done {

text-decoration: line-through;

color: #bdbdbd;

background: #f7f7f7;

}

.deleteBtn {

background: #1976d2;

color: #fff;

border: none;

padding: 7px 14px;

cursor: pointer;

border-radius: 5px;

font-size: 14px;

font-weight: 600;

transition: background 0.2s;

}

.deleteBtn:hover {

background: #0d47a1;

}

#jokeBtn {

padding: 10px 16px;

border-radius: 7px;

border: none;

background: linear-gradient(90deg, #2196f3 0%, #90caf9 100%);

color: #fff;

font-weight: 600;

cursor: pointer;

transition: background 0.2s, transform 0.2s;

}

#jokeBtn:hover {

background: linear-gradient(90deg, #90caf9 0%, #2196f3 100%);

color: #111;

transform: translateY(-2px) scale(1.04);

}

#jokeDisplay {

margin-top: 12px;

min-height: 40px;

font-size: 16px;

color: #333;

}

</style>

</head>

<body>

<h1>Interactive Web App</h1>

<div class="container">

<!-- Jokes Fetch Box -->

<div class="box" id="jokeBox">

<h2>Random Joke</h2>

<button id="jokeBtn">Get a Joke</button>

<div id="jokeDisplay"></div>

</div>

<!-- To-Do List Box -->

<div class="box">

<h2>To-Do List</h2>

<input type="text" id="taskInput" placeholder="Add new task">

<button id="addTaskBtn">Add Task</button>

<ul id="taskList"></ul>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

***script.js***

// --------- API FETCH FEATURE ---------

// ================== Jokes Fetch ==================

const jokeBtn = document.getElementById('jokeBtn');

const jokeDisplay = document.getElementById('jokeDisplay');

jokeBtn.addEventListener('click', () => {

jokeDisplay.innerHTML = '<em>Loading...</em>';

fetch('https://official-joke-api.appspot.com/random\_joke')

.then(res => res.json())

.then(joke => {

jokeDisplay.innerHTML = `<strong>${joke.setup}</strong><br><span style='color:#1976d2;'>${joke.punchline}</span>`;

})

.catch(err => {

jokeDisplay.innerHTML = '<em>Error fetching joke.</em>';

});

});

// ================== To-Do List ==================

const taskInput = document.getElementById('taskInput');

const addTaskBtn = document.getElementById('addTaskBtn');

const taskList = document.getElementById('taskList');

addTaskBtn.addEventListener('click', () => {

const taskText = taskInput.value.trim();

if(taskText === '') return;

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

li.textContent = taskText;

// Toggle done

li.addEventListener('click', () => li.classList.toggle('done'));

// Delete button

const deleteBtn = document.createElement('button');

deleteBtn.textContent = 'Delete';

deleteBtn.classList.add('deleteBtn');

deleteBtn.addEventListener('click', (e) => {

e.stopPropagation(); // Prevent marking as done

li.remove();

});

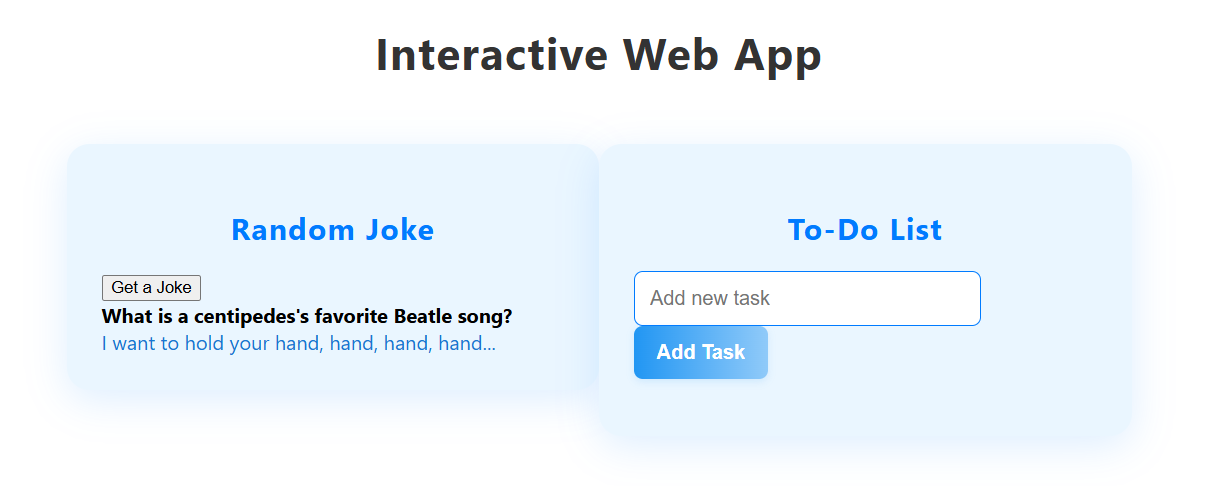
li.appendChild(deleteBtn);

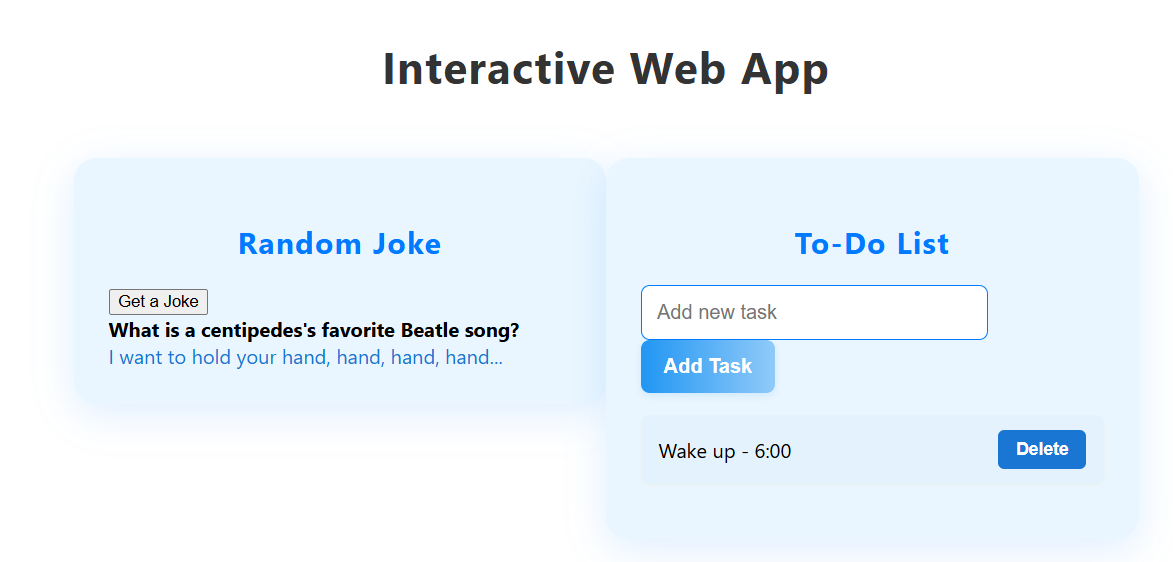
taskList.appendChild(li);

taskInput.value = '';

});

**Output**

****

****