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>API Fetch & To-Do List App</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background: #f9f9f9;

            margin: 0;

            padding: 20px;

        }

        h2 {

            color: #333;

        }

        .section {

            background: #fff;

            padding: 15px;

            margin-bottom: 25px;

            border-radius: 6px;

            box-shadow: 0 4px 10px rgba(0,0,0,0.05);

        }

        button {

            padding: 8px 15px;

            margin-top: 10px;

            border: none;

            background: #007bff;

            color: #fff;

            border-radius: 4px;

            cursor: pointer;

        }

        button:hover {

            background: #0056b3;

        }

        #apiData {

            margin-top: 10px;

            font-size: 1rem;

            color: #444;

        }

        ul {

            list-style: none;

            padding-left: 0;

        }

        li {

            padding: 8px;

            background: #f1f1f1;

            margin-bottom: 8px;

            border-radius: 4px;

            display: flex;

            justify-content: space-between;

            align-items: center;

        }

        li.completed {

            text-decoration: line-through;

            color: gray;

        }

        input {

            padding: 8px;

            margin-right: 10px;

            border: 1px solid #ccc;

            border-radius: 4px;

        }

    </style>

</head>

<body>

    <!-- API Fetch Section -->

    <div class="section">

        <h2>Fetch Random Joke</h2>

        <button id="fetchBtn">Get Joke</button>

        <div id="apiData">Click the button to fetch a joke...</div>

    </div>

    <!-- To-Do List Section -->

    <div class="section">

        <h2>My To Do List</h2>

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

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

        <ul id="todoList">

            <li>EduNet Foundation <button style="background: #dc3545; color: white; margin-left: 10px; cursor: pointer;">X</button></li>

        </ul>

    </div>

    <script src="app.js"></script>

</body>

</html>

***script.js***

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

const fetchBtn = document.getElementById("fetchBtn");

const apiDataDiv = document.getElementById("apiData");

fetchBtn.addEventListener("click", fetchJoke);

function fetchJoke() {

    // Fetching a random joke from an open API

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

        .then(response => response.json())

        .then(data => {

            apiDataDiv.textContent = `${data.setup} — ${data.punchline}`;

        })

        .catch(error => {

            apiDataDiv.textContent = "Failed to fetch joke.";

            console.error(error);

        });

}

// --------- TO-DO LIST FEATURE ---------

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

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

const todoList = document.getElementById("todoList");

addTaskBtn.addEventListener("click", addTask);

function addTask() {

    const taskText = taskInput.value.trim();

    if (taskText === "") {

        alert("Please enter a task.");

        return;

    }

    // Create list item

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

    li.textContent = taskText;

    // Mark complete on click

    li.addEventListener("click", () => {

        li.classList.toggle("completed");

    });

    // Delete button for each task

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

    deleteBtn.textContent = "X";

    deleteBtn.style.background = "#dc3545";

    deleteBtn.style.color = "white";

    deleteBtn.style.marginLeft = "10px";

    deleteBtn.style.cursor = "pointer";

    deleteBtn.onclick = () => li.remove();

    li.appendChild(deleteBtn);

    todoList.appendChild(li);

    // Clear input field

    taskInput.value = "";

}

// Initial setup for the existing task

document.querySelectorAll('#todoList li').forEach(item => {

    const deleteBtn = item.querySelector('button');

    deleteBtn.onclick = () => item.remove();

    item.addEventListener("click", () => {

        item.classList.toggle("completed");

    });

});

**Output**

