**Assignment 9: DOM Manipulation**

**Task: Build a to-do list application where users can add, edit, and delete tasks. Implement dynamic updates to the DOM based on user interactions.**

Below is an example of a to-do list application built using HTML, CSS, and JavaScript for DOM manipulation. Users can add, edit, and delete tasks with dynamic updates to the DOM.

HTML (index.html):

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

CSS (styles.css):

| body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  display: flex;  align-items: center;  justify-content: center;  height: 100vh;  background-color: #f5f5f5;  }  .container {  text-align: center;  }  input {  font-size: 16px;  padding: 8px;  }  button {  font-size: 16px;  padding: 8px 16px;  cursor: pointer;  background-color: #4caf50;  color: #fff;  border: none;  border-radius: 4px;  }  button:hover {  background-color: #45a049;  }  ul {  list-style-type: none;  padding: 0;  }  li {  font-size: 16px;  margin: 10px 0;  display: flex;  align-items: center;  }  .task-text {  flex-grow: 1;  margin-right: 10px;  }  .edit-button,  .delete-button {  font-size: 14px;  padding: 4px 8px;  cursor: pointer;  background-color: #e0e0e0;  color: #333;  border: none;  border-radius: 4px;  margin-left: 5px;  }  .edit-button:hover,  .delete-button:hover {  background-color: #cccccc;  } |
| --- |

JavaScript (script.js):

| const taskInput = document.getElementById('taskInput');  const addTaskButton = document.getElementById('addTaskButton');  const taskList = document.getElementById('taskList');  addTaskButton.addEventListener('click', addTask);  function addTask() {  const taskText = taskInput.value.trim();    if (taskText !== '') {  const li = document.createElement('li');  li.innerHTML = `  <span class="task-text">${taskText}</span>  <button class="edit-button">Edit</button>  <button class="delete-button">Delete</button>  `;    taskList.appendChild(li);  taskInput.value = '';  // Add event listeners for edit and delete buttons  const editButton = li.querySelector('.edit-button');  const deleteButton = li.querySelector('.delete-button');  editButton.addEventListener('click', () => editTask(li));  deleteButton.addEventListener('click', () => deleteTask(li));  }  }  function editTask(li) {  const taskTextElement = li.querySelector('.task-text');  const updatedText = prompt('Edit task:', taskTextElement.textContent.trim());  if (updatedText !== null) {  taskTextElement.textContent = updatedText;  }  }  function deleteTask(li) {  if (confirm('Are you sure you want to delete this task?')) {  li.remove();  }  } |
| --- |

In this example:

The HTML file includes an input field, an "Add Task" button, and an unordered list (<ul>) for displaying tasks.

The CSS file provides styling for the layout and appearance of the elements.

The JavaScript file handles the functionality of adding, editing, and deleting tasks. Event listeners are added to the "Add Task" button and dynamically created edit and delete buttons.

When a task is added, a new list item (<li>) is created and appended to the task list. Edit and delete buttons are added with event listeners for editing and deleting tasks.

The editTask function prompts the user to edit the task text, and the deleteTask function confirms if the user wants to delete the task.

To test this, open the index.html file in a web browser. Users can add, edit, and delete tasks with dynamic updates to the DOM.