Dynamic To-Do List

Coding exercise for beginners

Excited to share a fun and hands-on JavaScript exercise for beginners

Below is an advanced JavaScript exercise that involves creating a dynamic to-do list application. The exercise covers HTML for the structure, CSS for styling, and JavaScript for dynamic functionality. The goal is to allow users to add, remove, and mark tasks as completed.

Exercise: Dynamic To-Do List

Step 1: HTML Structure

Create the HTML structure for the to-do list. Include input fields for adding tasks, a list to display tasks, and buttons for interaction.

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

Step 2: CSS Styling

Create a simple CSS file to style the to-do list.

```
body {
  font-family: 'Arial', sans-serif;
  background-color: #f0f0f0;
  margin: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
}

.container {
  background-color: #fff;
  padding: 20px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  border-radius: 8px;
  width: 300px;
  text-align: center;
}

input {
  padding: 8px;
```

```
  margin-right: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  padding: 8px 16px;
  background-color: #4caf50;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

ul {
  list-style: none;
  padding: 0;
}

li {
  display: flex;
  justify-content: space-between;
  align-items: center;
```

```
  padding: 8px;
  border-bottom: 1px solid #ddd;
}

.completed {
  text-decoration: line-through;
  color: #888;
}
```

Step 3: JavaScript Functionality

Create the JavaScript file to handle the dynamic functionality of the to-do list.

```
// script.js

// Function to add a new task
function addTask() {
  const taskInput =
document.getElementById('taskInput');
  const taskList = document.getElementById('taskList');

  // Check if the input is not empty
  if (taskInput.value.trim() !== '') {
    // Create a new list item
    const listItem = document.createElement('li');
```

```
    // Create a span for the task text
    const taskText = document.createElement('span');
    taskText.textContent = taskInput.value;

    // Create buttons for marking as complete and
removing the task
    const completeButton =
document.createElement('button');
    completeButton.textContent = 'Complete';
    completeButton.onclick = function () {
      listItem.classList.toggle('completed');
    };

    const removeButton =
document.createElement('button');
    removeButton.textContent = 'Remove';
    removeButton.onclick = function () {
      listItem.remove();
    };

    // Append elements to the list item
    listItem.appendChild(taskText);
```

```
    listItem.appendChild(completeButton);
    listItem.appendChild(removeButton);

    // Append the list item to the task list
    taskList.appendChild(listItem);

    // Clear the input field
    taskInput.value = '';
  }
}
```

Explanation:

- The HTML structure contains input fields, buttons, and a list to display tasks.
- CSS provides simple styling for the to-do list, making it visually appealing.
- JavaScript handles the dynamic functionality:
    - The addTask function adds a new task to the list when the "Add Task" button is clicked.
    - Each task has buttons for marking as complete and removing the task.
    - The completed class is toggled to apply a strike-through effect when a task is marked as complete.
    - The remove button removes the respective task from the list.

Instructions for Users:

1. Type a task in the input field.
2. Click "Add Task" to add the task to the list.
3. Click "Complete" to mark a task as complete (adds a strike-through).
4. Click "Remove" to remove a task from the list.

This exercise helps beginners understand how to manipulate the DOM using JavaScript to create dynamic and interactive web applications. It covers basic HTML structure, CSS styling, and JavaScript event handling.