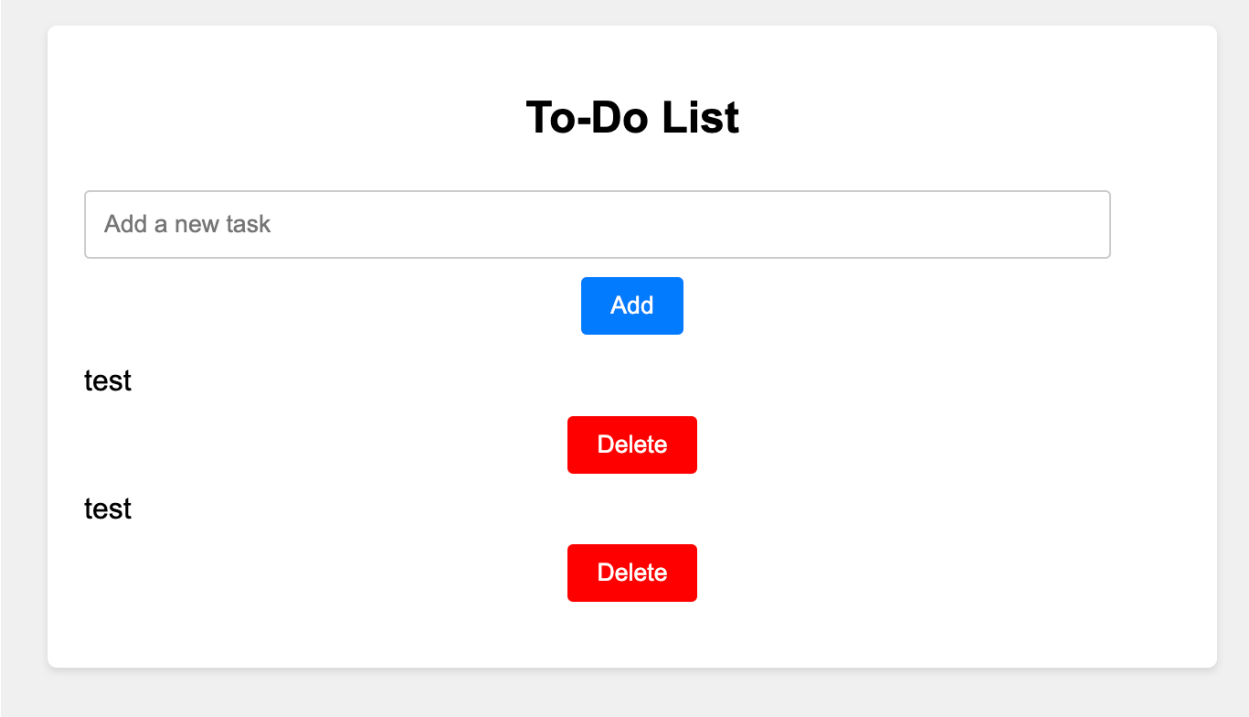
JavaScript code Example ToDo List

JavaScript project that creates a basic to-do list application



Here's a simple JavaScript project that creates a basic to-do list application. I'll provide you with the step-by-step description and the full code for each step.

Project: Simple To-Do List App

Step 1: HTML Structure

Create an HTML file named index.html and set up the basic structure.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Simple To-Do List</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="addButton">Add</button>
        <ul id="taskList"></ul>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Step 2: CSS Styling

Create a CSS file named styles.css for basic styling.

```
body {
```

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

.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    width: 300px;
}

h1 {
    font-size: 24px;
    text-align: center;
}

input[type="text"] {
```

```
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    border: 1px solid #ccc;
    border-radius: 3px;
}

button {
    display: block;
    margin: 10px auto;
    padding: 8px 16px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 3px;
    cursor: pointer;
}

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

Step 3: JavaScript Logic

Create a JavaScript file named script.js for the application logic.

```
const taskInput = document.getElementById("taskInput");
const addButton = document.getElementById("addButton");
const taskList = document.getElementById("taskList");

addButton.addEventListener("click", addTask);

function addTask() {
    const taskText = taskInput.value.trim();

    if (taskText !== "") {
        const li = document.createElement("li");
        li.textContent = taskText;

        const deleteButton =
document.createElement("button");
        deleteButton.textContent = "Delete";
        deleteButton.addEventListener("click", () => {
            taskList.removeChild(li);
        });

        li.appendChild(deleteButton);
```

```
        taskList.appendChild(li);

        taskInput.value = "";
    }
}
```

Step 4: Testing

Open the index.html file in a web browser. You should see the simple to-do list application with an input field and an "Add" button. Enter tasks and click the "Add" button to add them to the list. Each task will have a "Delete" button to remove it from the list.

JavaScript Code Details

```
// Get references to the HTML elements
const taskInput = document.getElementById("taskInput");
const addButton = document.getElementById("addButton");
const taskList = document.getElementById("taskList");

// Attach an event listener to the "Add" button
addButton.addEventListener("click", addTask);

// Define the function to add a task
```

```
function addTask() {
    // Get the trimmed value from the task input field
    const taskText = taskInput.value.trim();

    // Check if the task is not an empty string
    if (taskText !== "") {
        // Create a new <li> element to represent the
task
        const li = document.createElement("li");
        li.textContent = taskText;

        // Create a "Delete" button for each task
        const deleteButton =
document.createElement("button");
        deleteButton.textContent = "Delete";

        // Attach an event listener to the "Delete"
button
        deleteButton.addEventListener("click", () => {
            // Remove the task's corresponding <li>
element
            taskList.removeChild(li);
        });
```

```
        // Append the "Delete" button to the task's
<li> element
        li.appendChild(deleteButton);

        // Append the task's <li> element to the task
list
        taskList.appendChild(li);

        // Clear the input field after adding the task
        taskInput.value = "";
    }
}
```

Step by Step Explanation:

1. We start by getting references to the HTML elements we'll be interacting with: taskInput, addButton, and taskList. These are obtained using the getElementById method, which fetches an element by its ID attribute.
2. We attach an event listener to the "Add" button (addButton). This listener is set to call the addTask function whenever the button is clicked.
3. The addTask function is defined. This function is responsible for adding a new task to the list.

4. Inside the addTask function, we retrieve the trimmed value from the task input field using taskInput.value.trim(). Trimming removes any leading or trailing spaces, ensuring we don't add empty tasks.
5. We check if the trimmed task text is not an empty string. If it's not empty, we proceed to create the HTML structure for the new task.
6. We create a new <li> element using document.createElement("li") to represent the task. We set its textContent to the value of the task input.
7. We create a "Delete" button for each task using document.createElement("button"). We set its textContent to "Delete".
8. We attach an event listener to the "Delete" button using deleteButton.addEventListener("click", ...). Inside the listener, we use the taskList.removeChild(li) method to remove the task's corresponding <li> element when the "Delete" button is clicked.
9. We append the "Delete" button to the task's <li> element using li.appendChild(deleteButton).
10. We append the task's <li> element, including the "Delete" button, to the task list (taskList.appendChild(li)).
11. After adding the task to the list, we clear the task input field by setting its value property to an empty string (taskInput.value = "").

By following these steps, the code creates a simple to-do list application where you can add tasks and delete them using the "Delete" button. This is a basic example, and you can build upon it by adding more features and functionality to create a more robust to-do list application.