Assignment-4

To-Do List Using HTML, CSS, and JavaScript

Presented by: **Hemish.P**
Subject/Course: **Web Development**
Date: **04/06/2025**

Introduction

- A To-Do List is a digital checklist that helps manage tasks.
- It helps improve productivity, organization, and time management.
- Created using HTML, CSS, and JavaScript in a single file.

Working

1. User enters a task into an input field.
2. On clicking "Add Task", the task appears in a list.
3. Each task can be individually removed.
4. JavaScript handles all interactivity and logic.

Uses

- Helps students manage assignments and deadlines.
- Useful for professionals to organize daily tasks.
- Can be expanded into project management tools.
- Simple tool for personal productivity.

Advantages

- Easy to use and lightweight.
- Improves personal organization.
- Fully browser-based – no installations needed.
- Can be enhanced with more features (e.g., storage, priority tags).

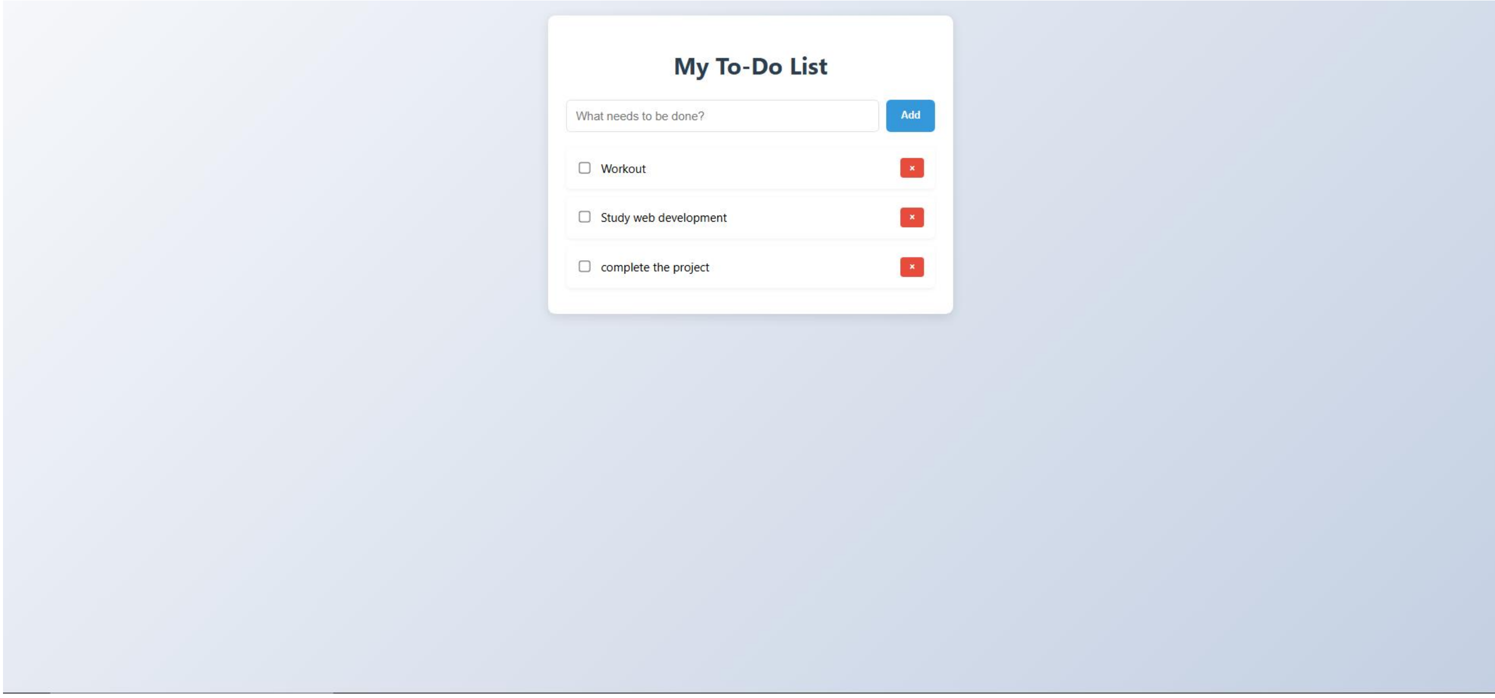
Disadvantages

- No data persistence (tasks are lost on refresh).
- Limited to basic functionality.
- Not suitable for complex project tracking.
- Lacks user authentication or multi-user support.

Future Scope

- Add Local Storage or Database Integration.
- Task priority levels and due dates.
- User authentication (login systems).
- Mobile app version using frameworks like React Native.
- Integration with calendars or notifications.

Output

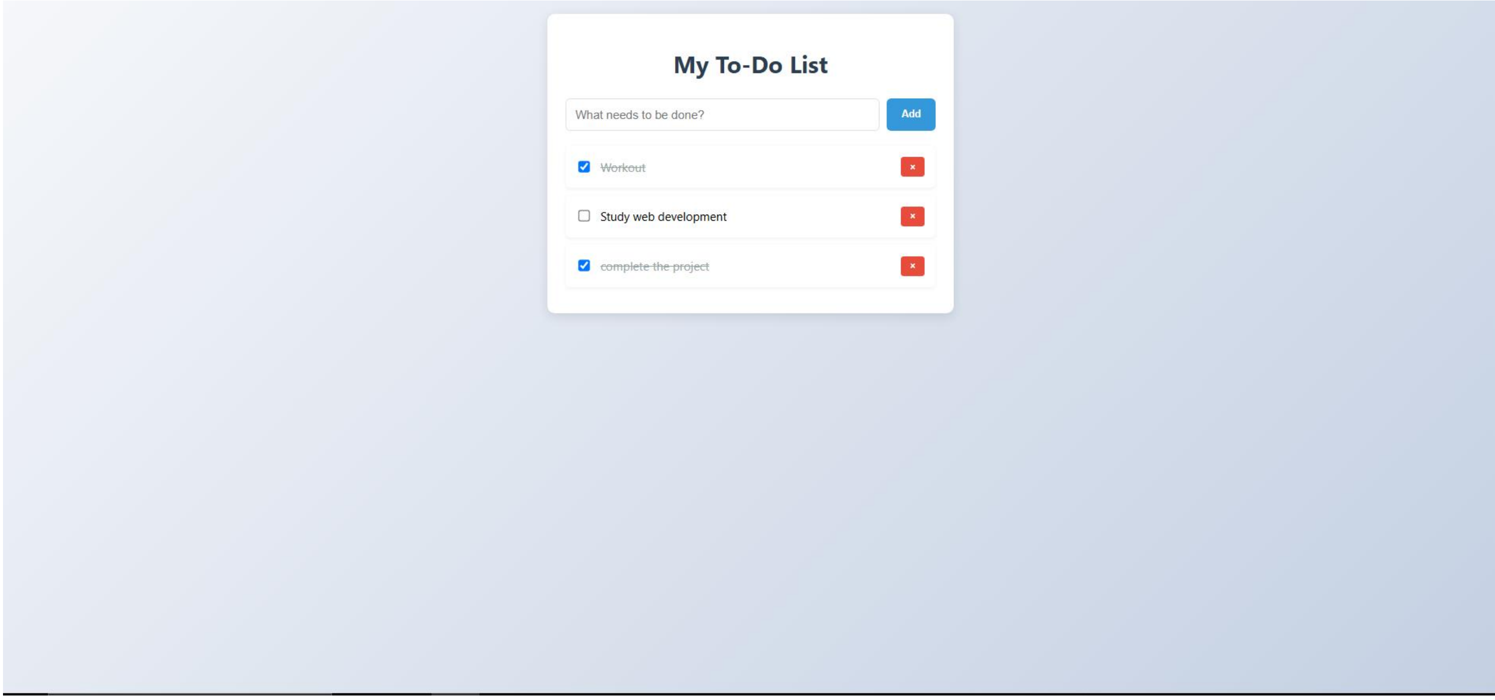


My To-Do List

☐ Workout ×

☐ Study web development ×

☐ complete the project ×



My To-Do List

What needs to be done?

☑ ~~Workout~~

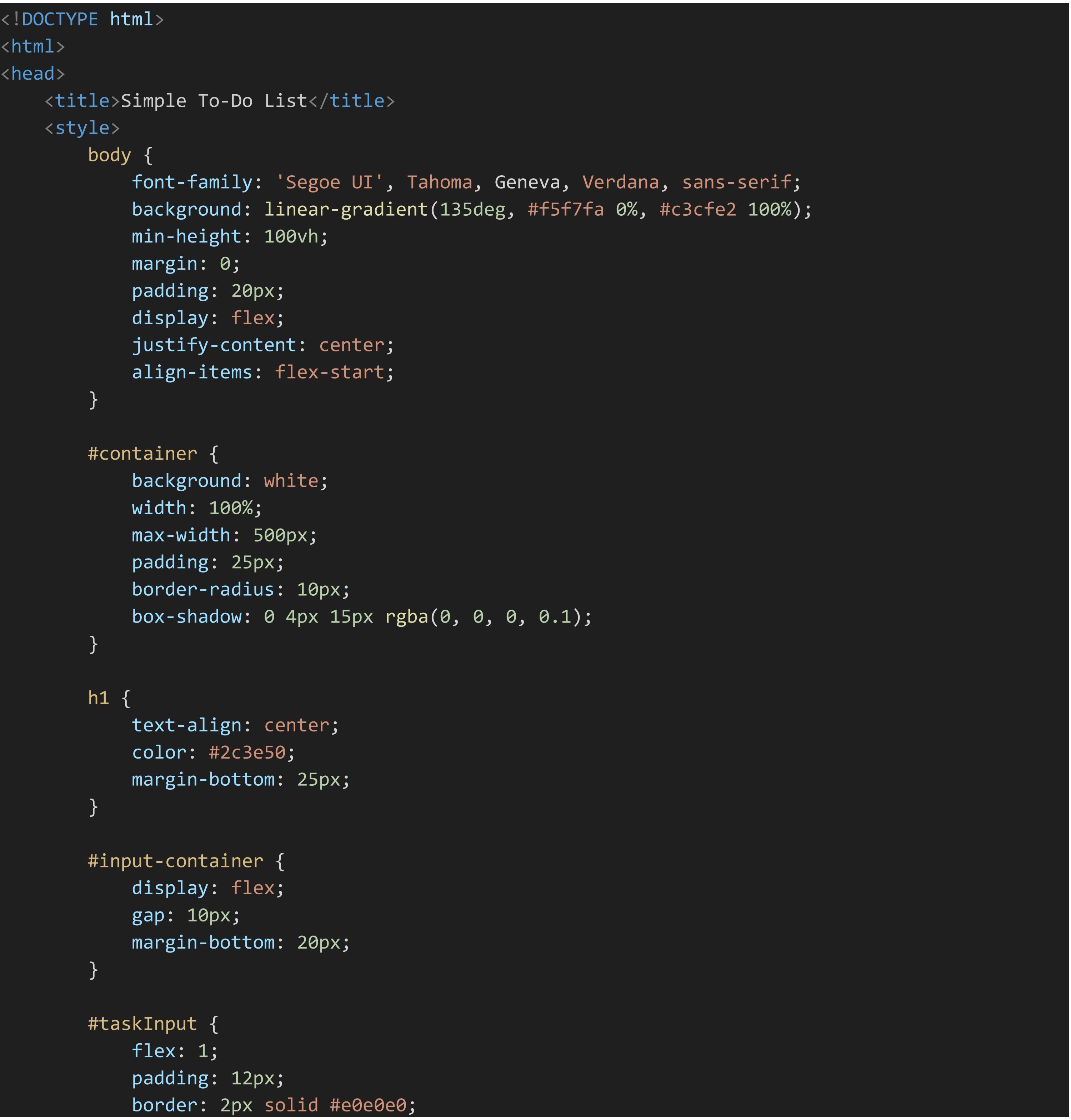
☐ Study web development

☑ ~~complete the project~~

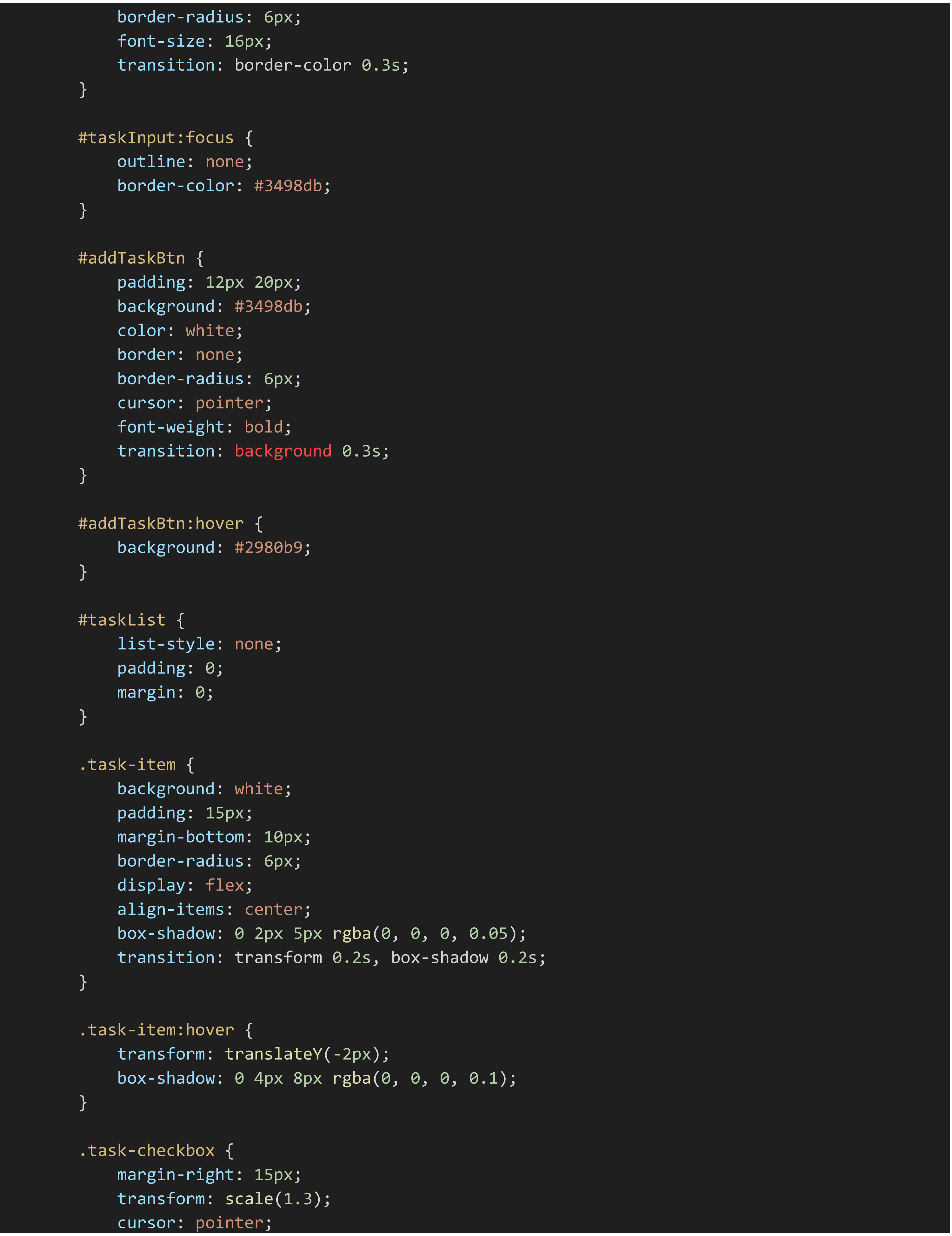
Conclusion

- A To-Do List is a great example of using front-end technologies to solve real-life problems.
- It is simple yet powerful in improving productivity.
- Offers a strong base for learning JavaScript and DOM manipulation.
- Easily expandable into a full-featured productivity tool.

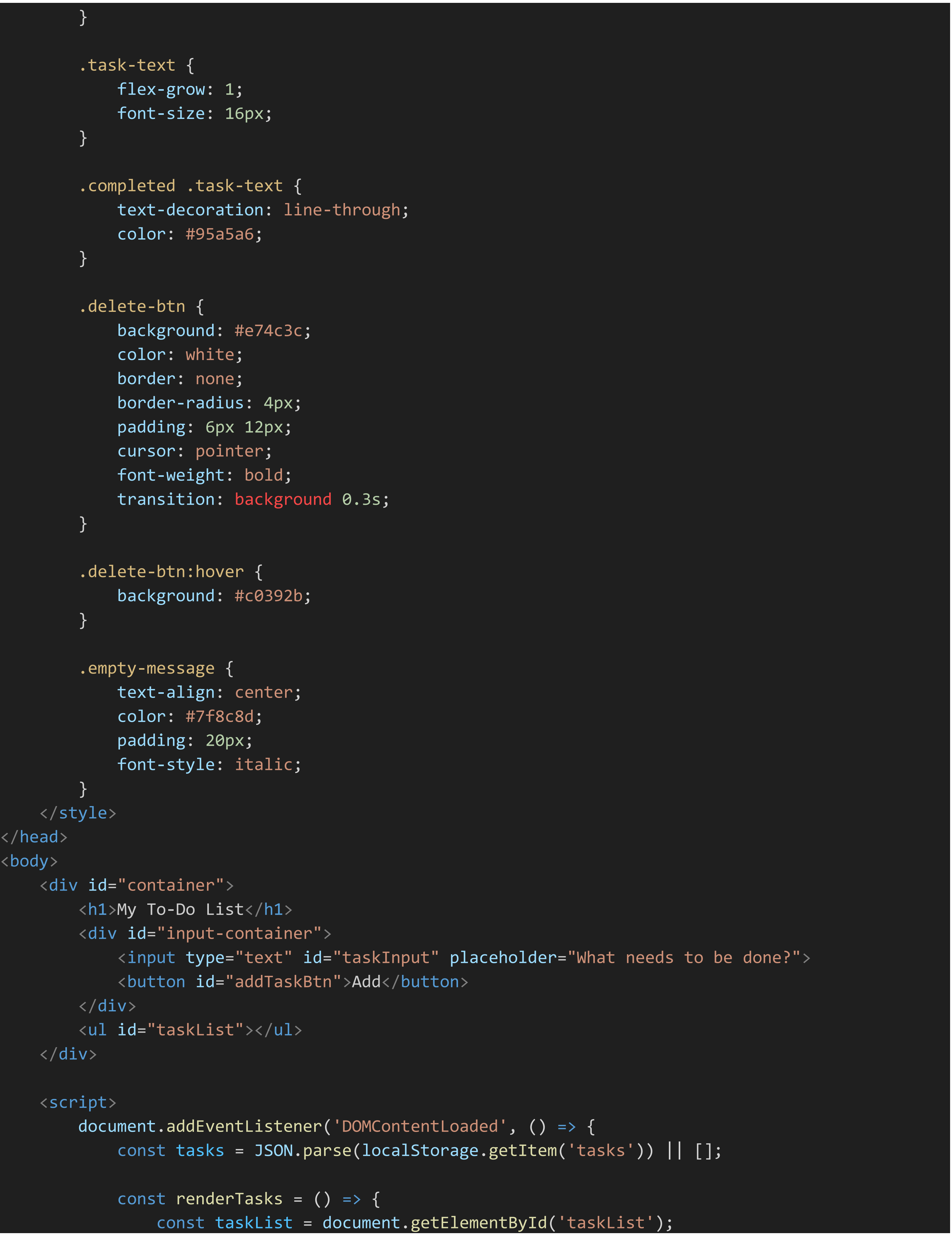
Source code



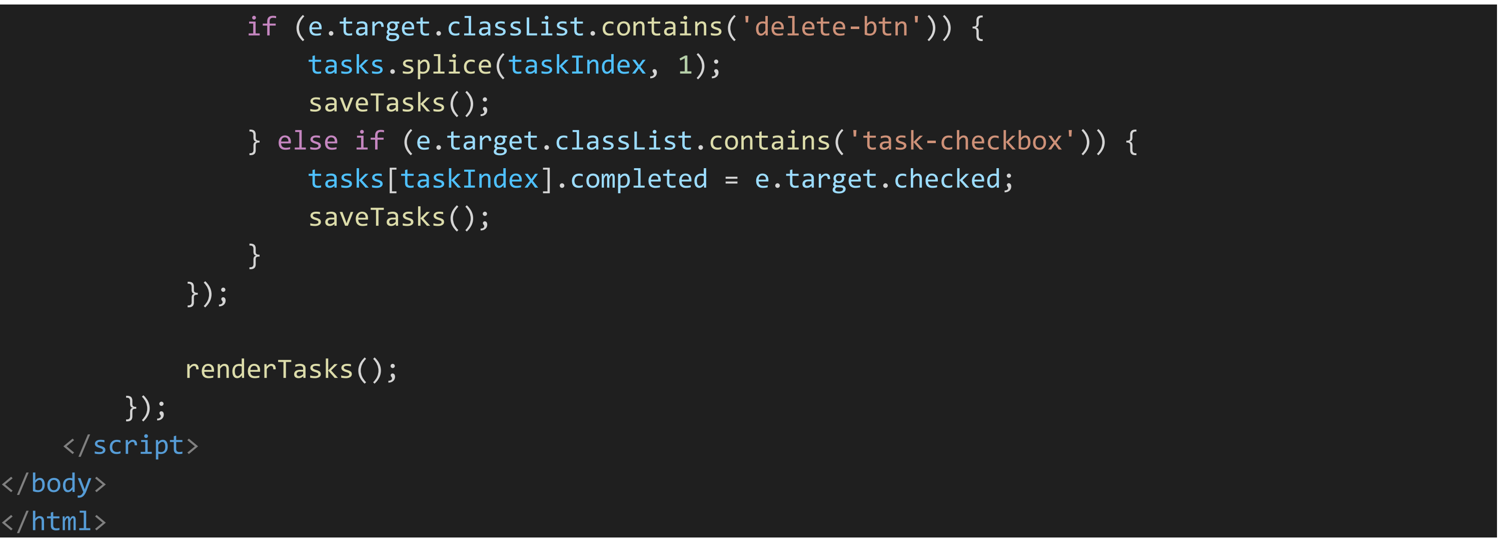
```
<!DOCTYPE html>
<html>
<head>
    <title>Simple To-Do List</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
            min-height: 100vh;
            margin: 0;
            padding: 20px;
            display: flex;
            justify-content: center;
            align-items: flex-start;
        }

        #container {
            background: white;
            width: 100%;
            max-width: 500px;
            padding: 25px;
            border-radius: 10px;
            box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            color: #2c3e50;
            margin-bottom: 25px;
        }

        #input-container {
            display: flex;
            gap: 10px;
            margin-bottom: 20px;
        }

        #taskInput {
            flex: 1;
            padding: 12px;
            border: 2px solid #e0e0e0;
```



```
        border-radius: 6px;
        font-size: 16px;
        transition: border-color 0.3s;
    }

    #taskInput:focus {
        outline: none;
        border-color: #3498db;
    }

    #addTaskBtn {
        padding: 12px 20px;
        background: #3498db;
        color: white;
        border: none;
        border-radius: 6px;
        cursor: pointer;
        font-weight: bold;
        transition: background 0.3s;
    }

    #addTaskBtn:hover {
        background: #2980b9;
    }

    #taskList {
        list-style: none;
        padding: 0;
        margin: 0;
    }

    .task-item {
        background: white;
        padding: 15px;
        margin-bottom: 10px;
        border-radius: 6px;
        display: flex;
        align-items: center;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
        transition: transform 0.2s, box-shadow 0.2s;
    }

    .task-item:hover {
        transform: translateY(-2px);
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .task-checkbox {
        margin-right: 15px;
        transform: scale(1.3);
        cursor: pointer;
```



```
        }

        .task-text {
            flex-grow: 1;
            font-size: 16px;
        }

        .completed .task-text {
            text-decoration: line-through;
            color: #95a5a6;
        }

        .delete-btn {
            background: #e74c3c;
            color: white;
            border: none;
            border-radius: 4px;
            padding: 6px 12px;
            cursor: pointer;
            font-weight: bold;
            transition: background 0.3s;
        }

        .delete-btn:hover {
            background: #c0392b;
        }

        .empty-message {
            text-align: center;
            color: #7f8c8d;
            padding: 20px;
            font-style: italic;
        }
    </style>
</head>
<body>
    <div id="container">
        <h1>My To-Do List</h1>
        <div id="input-container">
            <input type="text" id="taskInput" placeholder="What needs to be done?">
            <button id="addTaskBtn">Add</button>
        </div>
        <ul id="taskList"></ul>
    </div>

    <script>
        document.addEventListener('DOMContentLoaded', () => {
            const tasks = JSON.parse(localStorage.getItem('tasks')) || [];

            const renderTasks = () => {
                const taskList = document.getElementById('taskList');
                if (tasks.length === 0) {
                    taskList.innerHTML = '<li class="empty-message">No tasks yet. Add one
above!</li>';
                    return;
                }

                taskList.innerHTML = tasks.map(task => `
                    <li class="task-item ${task.completed ? 'completed' : ''}" data-
id="${task.id}">
                        <input type="checkbox" class="task-checkbox" ${task.completed ?
'checked' : ''}>
                        <span class="task-text">${task.text}</span>
                        <button class="delete-btn">×</button>
                    </li>
                `).join('');
            };

            const saveTasks = () => {
                localStorage.setItem('tasks', JSON.stringify(tasks));
                renderTasks();
            };

            document.getElementById('addTaskBtn').addEventListener('click', () => {
                const input = document.getElementById('taskInput');
                const text = input.value.trim();

                if (text) {
                    tasks.push({
                        id: Date.now(),
                        text,
                        completed: false
                    });
                    input.value = '';
                    saveTasks();
                }
            });

            document.getElementById('taskInput').addEventListener('keypress', (e) => {
                if (e.key === 'Enter') {
                    document.getElementById('addTaskBtn').click();
                }
            });

            document.getElementById('taskList').addEventListener('click', (e) => {
                const taskItem = e.target.closest('.task-item');
                if (!taskItem) return;

                const taskId = Number(taskItem.dataset.id);
                const taskIndex = tasks.findIndex(t => t.id === taskId);
```



```
                if (e.target.classList.contains('delete-btn')) {
                    tasks.splice(taskIndex, 1);
                    saveTasks();
                } else if (e.target.classList.contains('task-checkbox')) {
                    tasks[taskIndex].completed = e.target.checked;
                    saveTasks();
                }
            });

            renderTasks();
        });
    </script>
</body>
</html>
```