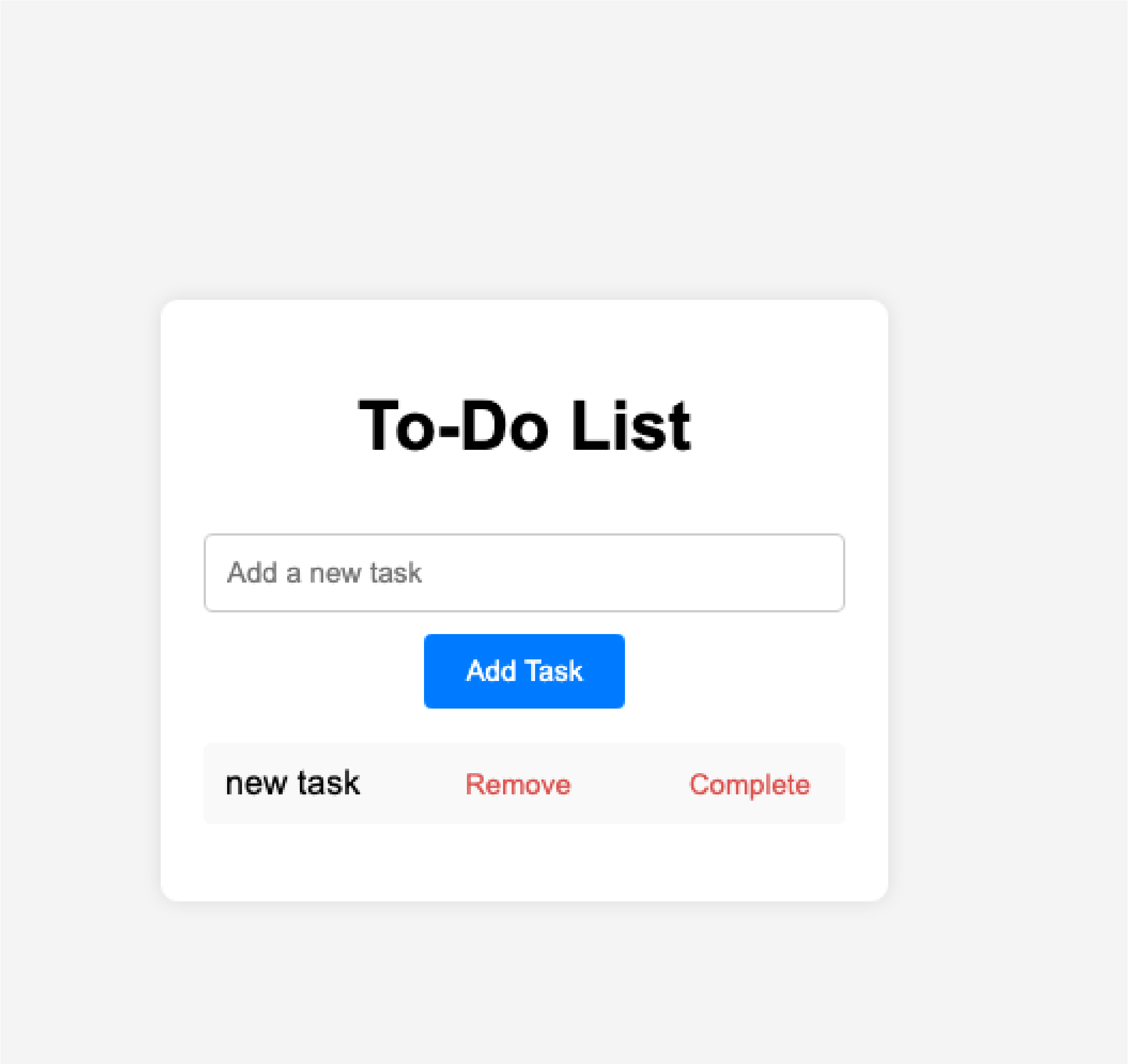
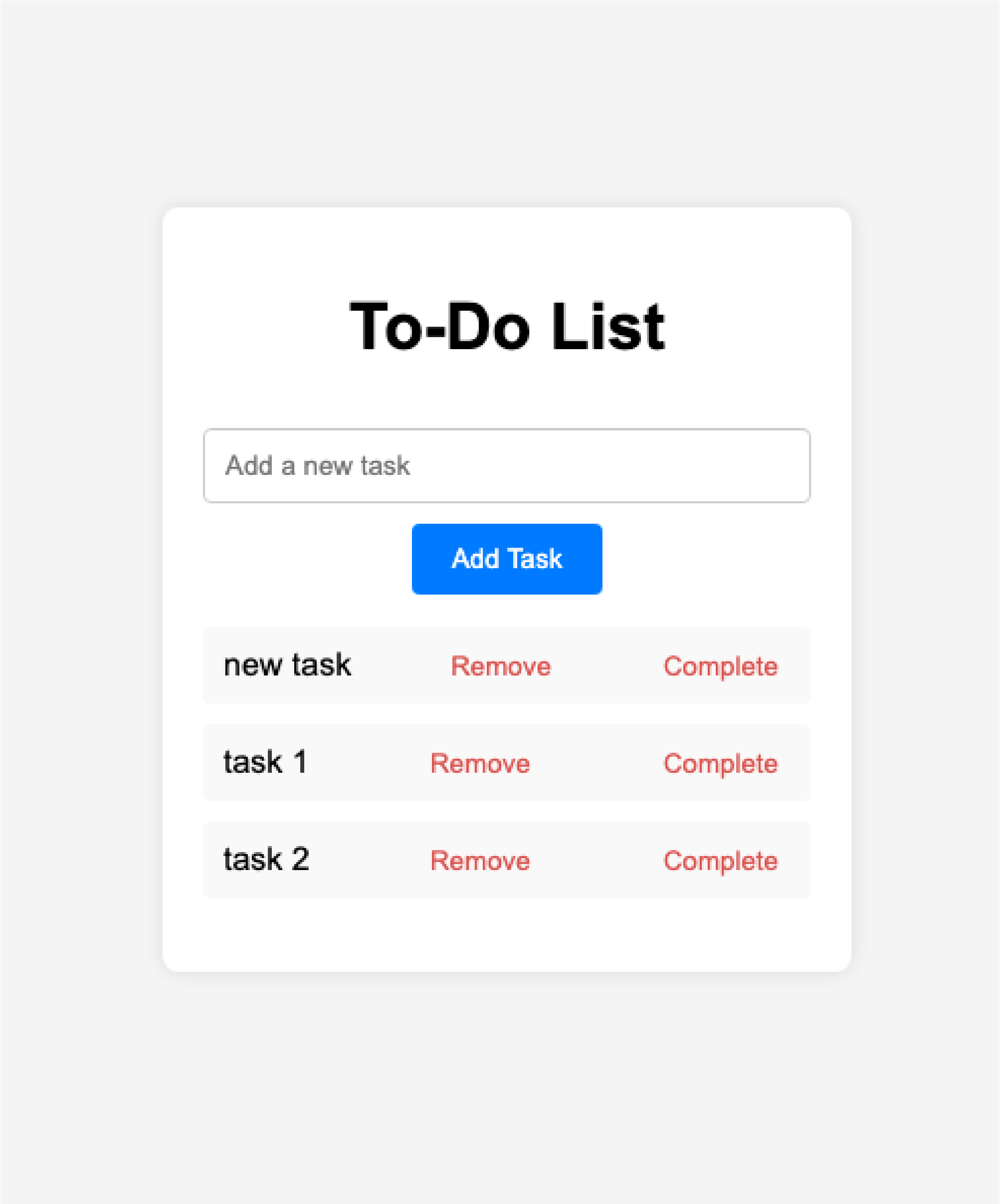
Week 15 Exercise.

Using JS to perform tasks like the following screen:





Answer the following questions: (Write "answer" below each question)

1. List CSS component to control position of the box

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}
.todo-container {
    width: 400px;
    background-color: #f5f5f5;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

2. List CSS component to add button

```
button {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    width: 100%;
    margin-bottom: 10px;
}
button.remove-btn, button.complete-btn {
    background-color: transparent;
    color: red;
    border: none;
    cursor: pointer;
    font-size: 14px;
    padding: 5px 10px;
}
```

3. List CSS component to add “list of tasks”

```
ul {
    list-style: none;
    padding: 0;
}
li {
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid #ccc;
    padding: 10px 0;
}
li:last-child {
    border-bottom: none;
}
```

4. List all JS variable and constants. Provide name and its purpose in your program

- taskInput: A variable that holds a reference to the input field where users type new tasks. It is used to read the value entered by the user.
- addButton: A variable that holds a reference to the button that triggers the addition of a new task to the list when clicked.
- taskList -different: A constant that holds a reference to the element containing the task list. It is used to append new task items to the list and to remove them from the list.<ul>

5. List at least one control structure. Suggest one alternative structure to achieve the same purpose.

```
function addTask() {
    try {
        const taskText = taskInput.value.trim();
        if (taskText === '') {
            throw new Error('请输入任务内容！');
        }
        const li = document.createElement('li');
        li.innerHTML = `
            <span>${taskText}</span>
            <button class="remove-btn">Remove</button>
            <button class="complete-btn">Complete</button>
        `;
        const removeBtn = li.querySelector('.remove-btn');
        const completeBtn = li.querySelector('.complete-btn');

        removeBtn.addEventListener('click', function() {
            taskList.removeChild(li);
        });

        completeBtn.addEventListener('click', function() {
            li.classList.toggle('completed');
        });

        taskList.appendChild(li);
        taskInput.value = '';
    } catch (error) {
        alert(error.message);
    }
}
```

6. List one JS event available in your program. Explain its purpose.

```
// 监听回车键，按下回车键时也添加任务
taskInput.addEventListener('keyup', function(event) {
    if (event.key === 'Enter') {
        addTask();
    }
});
```

7. Explain the relevance of DOM model to your JS. List at least 3 relevancies.

The DOM model is crucial in the code for several reasons:

It enables interaction between JavaScript and HTML elements, allowing element access.

It facilitates dynamic creation, modification, and deletion of elements, updating the page in real-time.

It supports event listening, which helps implement interactive features in response to user actions.

It provides methods to manipulate element classes and properties, controlling appearance and state (e.g., marking tasks as completed).

Provide your html, CSS and JS.

Save this doc as pdf, and copy and paste to online text