
Page 1: HTML Structure and Basic Layout

Objective: Create the basic HTML structure for the to-do list, including input fields, buttons, and task containers.

Task Details:

1. **HTML Structure:**
    - **Task Input Section:**
        - Add an input field for entering new tasks, with a placeholder text like "Add a new task…".
        - Include an "Add" button next to the input field.
    - **Task List Section:**
        - Add a container `<ul>` or `<div>` to display the task list.
        - Each task will be represented by a `<li>` or `<div>` with the task text and two buttons: "Complete" and "Delete".

Structure:

```
 <div class="task-container">
  <input type="text" id="task-input" placeholder="Add a new task...">
  <button id="add-task-button">Add</button>
</div>
<ul id="task-list">
  <!-- Tasks will be dynamically added here -->
</ul>
```

    -
2. **CSS Basics:**
    - Set the background color to **white** (`#ffffff`).

Style the input field and button with basic styles:

```
 #task-input {
  padding: 8px;
  font-size: 16px;
  margin-right: 10px;
}
#add-task-button {
  background-color: #4CAF50;
  color: white;
  padding: 10px;
  border: none;
```

```
  cursor: pointer;
}
```

-

Expected Deliverables:

- Basic HTML structure for the task input and list, with styling for input fields and buttons.

Page 2: Task Functionality and Basic JavaScript

Objective: Add JavaScript logic for adding tasks, marking them as complete, and deleting them.

Task Details:

1. **Task Adding Functionality:**
    - Write a function to add tasks to the task list when the user presses the "Add" button or hits the Enter key.
    - Append a new task to the list dynamically. Each task should have a "Complete" and "Delete" button.

Example:

```
 const addTaskButton = document.getElementById('add-task-button');
addTaskButton.addEventListener('click', function() {
  const taskText = document.getElementById('task-input').value;
  if (taskText) {
    addTask(taskText);
    document.getElementById('task-input').value = '';
  }
});

function addTask(taskText) {
  const taskList = document.getElementById('task-list');
  const taskItem = document.createElement('li');
  taskItem.innerHTML = `${taskText} <button class="complete">Complete</button> <button
class="delete">Delete</button>`;
  taskList.appendChild(taskItem);
}
```

-

2. **Task Removal and Completion:**

- Add functionality to delete tasks when the "Delete" button is clicked.
- Add functionality to mark tasks as complete when the "Complete" button is clicked, applying a visual change (e.g., strikethrough or color change).

Example:

```
 document.addEventListener('click', function(e) {
  if (e.target.classList.contains('delete')) {
    e.target.parentElement.remove();
  }
  if (e.target.classList.contains('complete')) {
    e.target.parentElement.classList.toggle('completed');
  }
});
```

3. **CSS for Task Completion:**

Style completed tasks with a strikethrough or change in background color:

```
 .completed {
  text-decoration: line-through;
  background-color: #f0f0f0;
}
```

Expected Deliverables:

- JavaScript functions to add, delete, and mark tasks as completed. Styling for completed tasks.

Page 3: Responsive Design and Final Touches

Objective: Ensure the layout is responsive, tasks stack vertically on small screens, and input fields adjust size.

Task Details:

1. **Responsive Design:**
    - Use **media queries** to adjust the layout for smaller screen sizes, ensuring tasks stack vertically.

Example:

```
@media screen and (max-width: 600px) {
 #task-input {
  width: 80%;
  margin: 10px 0;
 }
 #add-task-button {
  width: 100%;
  padding: 12px;
 }
 #task-list {
  padding: 0;
  list-style-type: none;
 }
 #task-list li {
  padding: 10px;
  margin: 5px 0;
 }
}
```

2. **Button Hover Effects:**
    - Add slight hover effects on the "Complete" and "Delete" buttons to enhance interactivity.

Example:

```
button:hover {
 background-color: #45a049; /* Slight change in color on hover */
}
```

3. **Test Responsiveness:**
    - Ensure the to-do list works smoothly on mobile devices and larger screens.
    - Adjust task input field size and task layout to look good on both desktop and mobile.

Expected Deliverables:

- Fully responsive layout with tasks stacked on small screens.
- Interactive buttons with hover effects and input adjustments for smaller screens.
