Lab Exercise: Building a Simple To-Do App Using TypeScript and HTML

Objective:
The goal of this lab exercise is to build a simple To-Do app using TypeScript and HTML. This will help you practice essential TypeScript features such as types, interfaces, classes, modules, and DOM manipulation. By the end of the lab, you should have a functional To-Do app where users can add, delete, and mark tasks as completed.

Prerequisites

1. Basic understanding of TypeScript.
2. Familiarity with HTML and CSS.
3. Code editor (e.g., VSCode).
4. TypeScript compiler installed (npm install -g typescript).

Lab Setup

1. **Create Project Directory:**
   - Create a new directory for your project, e.g., todo-app.
   - Navigate to this directory in your terminal.
2. **Initialize TypeScript:**
   - Run tsc --init to create a tsconfig.json file in your project directory.
   - This file will allow you to configure TypeScript options.
3. **Set Up HTML File:**
   - Create an index.html file in the root of your project directory.
   - Add the following basic structure to your HTML file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>To-Do App</title>
  <style>
```

```
        /* Add some basic styles */
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .todo-list {
            list-style-type: none;
            padding: 0;
        }
        .todo-item {
            display: flex;
            justify-content: space-between;
            margin-bottom: 10px;
        }
        .completed {
            text-decoration: line-through;
        }
    </style>
</head>
<body>
    <h1>To-Do List</h1>
    <input type="text" id="new-task-input" placeholder="Add a new task..." />
    <button id="add-task-button">Add Task</button>
    <ul id="todo-list" class="todo-list"></ul>

    <script src="dist/app.js"></script>
</body>
</html>
```

4. **Set Up TypeScript File:**
    - Create a src directory inside your project directory.
    - Inside the src directory, create a file called app.ts.

5. **Compile TypeScript:**
    - Run tsc -w in your terminal to watch and compile TypeScript files automatically.

Task 1: Define the Task Interface

1. **Create an Interface for Tasks:**
    - Define an interface Task in your app.ts file with the following properties:

```
interface Task {
  id: number;
  title: string;
  completed: boolean;
}
```

2. **Explain the Interface:**
    - id: A unique identifier for each task.
    - title: The description of the task.
    - completed: A boolean indicating whether the task is completed.

Task 2: Create the To-Do Class

1. **Create a Class for Managing Tasks:**
    - Create a TodoApp class that will manage the list of tasks.
    - The class should have the following methods:
        - addTask(title: string): void
        - deleteTask(id: number): void
        - toggleTaskCompletion(id: number): void
        - renderTasks(): void
2. **Implement the Class:**

```
class TodoApp {
  private tasks: Task[] = [];
  private nextId: number = 1;

  addTask(title: string): void {
    const newTask: Task = { id: this.nextId++, title, completed: false };
    this.tasks.push(newTask);
    this.renderTasks();
}

deleteTask(id: number): void {
    this.tasks = this.tasks.filter(task => task.id !== id);
    this.renderTasks();
}

toggleTaskCompletion(id: number): void {
    const task = this.tasks.find(task => task.id === id);
    if (task) {
        task.completed = !task.completed;
        this.renderTasks();
    }
}

renderTasks(): void {
    const todoList = document.getElementById('todo-list') as HTMLUListElement;
    todoList.innerHTML = '';

    this.tasks.forEach(task => {
        const li = document.createElement('li');
        li.className = 'todo-item';

        const titleSpan = document.createElement('span');
        titleSpan.textContent = task.title;
        titleSpan.className = task.completed ? 'completed' : '';

        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Delete';
```

```
            deleteButton.onclick = () => this.deleteTask(task.id);

            const toggleButton = document.createElement('button');
            toggleButton.textContent = task.completed ? 'Undo' : 'Complete';
            toggleButton.onclick = () => this.toggleTaskCompletion(task.id);

            li.appendChild(titleSpan);
            li.appendChild(deleteButton);
            li.appendChild(toggleButton);

            todoList.appendChild(li);
        });
    }
}
```

Task 3: Integrate with HTML

1. **Connect the UI with TypeScript:**
    - In the app.ts file, add the following code to handle user input:

```
const todoApp = new TodoApp();

const addTaskButton = document.getElementById('add-task-button') as HTMLButtonElement;
const newTaskInput = document.getElementById('new-task-input') as HTMLInputElement;

addTaskButton.onclick = () => {
    const taskTitle = newTaskInput.value.trim();
    if (taskTitle) {
        todoApp.addTask(taskTitle);
        newTaskInput.value = '';
    }
};
```

2. **Explanation:**
    - When the "Add Task" button is clicked, the app will retrieve the input value, add a new task using the TodoApp class, and then clear the input field.
3. **Compile and Run:**
    - Ensure your TypeScript code is compiled by running tsc.
    - Open the index.html file in your browser to see the To-Do app in action.

Task 4: Advanced Features (Optional)

1. **Add Local Storage Support:**
    - Modify the TodoApp class to save tasks to and load tasks from localStorage so that the tasks persist between page reloads.
2. **Implement Task Editing:**
    - Allow users to edit the title of existing tasks.
3. **Filter Tasks:**
    - Add buttons to filter the list to show "All", "Completed", or "Pending" tasks.

Task 5: Code Review and Submission

1. **Review Your Code:**
    - Ensure your code follows TypeScript best practices.
    - Check for any unused variables or redundant code.
2. **Submit Your Project:**
    - Compress your project folder (including index.html, src/app.ts, and compiled dist/app.js) and submit it as instructed.

Conclusion

This lab exercise has guided you through building a simple To-Do app using TypeScript and HTML. By completing this exercise, you've practiced essential TypeScript features and learned how to integrate TypeScript with HTML for web development.