Building a Simple React Todo App: Step-by-Step Lab

Basel Magableh

24/3/2025

Session Goal (120 Minutes)

By the end of this guided lab, you will have:

- Installed a complete React development environment
- Built a small interactive Todo app
- Used components, state, props, and event handling
- Extended the app with extra features and challenges

Part 1 – Preparing Your Development Environment (15 min)

To develop with React, you'll need to set up the following:

1. Install Node.js and npm

Go to `https://nodejs.org/` and download the **LTS version**. npm comes bundled with Node.js.

2. Install Visual Studio Code

Visit `https://code.visualstudio.com/` and install VS Code.

3. Create a New React Project

Open a terminal and run:

```
npx create-react-app todo-app
cd todo-app
npm start
```

Your app should open in a browser at `http://localhost:3000`.

Part 2 – Build the Basic Todo App (30 min)

Student Task:

Replace the contents of `App.js` with the following starter component:

```
import React, { useState } from 'react';

function App() {
  const [todos, setTodos] = useState([]);
  const [input, setInput] = useState('');

  const handleAddTodo = (e) => {
    e.preventDefault();
    if (!input) return;
    setTodos([...todos, input]);
    setInput('');
  };

  const handleDeleteTodo = (index) => {
    const newTodos = todos.filter((_, i) => i !== index);
    setTodos(newTodos);
  };

  return (
    <div>
      <h2>Todo App</h2>
      <form onSubmit={handleAddTodo}>
        <input
          type="text"
          value={input}
          onChange={(e) => setInput(e.target.value)}
        />
        <button type="submit">Add Todo</button>
      </form>

      <ul>
        {todos.map((todo, index) => (
          <li key={index}>
            {todo}
            <button onClick={() => handleDeleteTodo(index)}>Delete</
               button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

Part 3 – How the App Works (15 min)

1. Add Todos

- Controlled input is tracked by `input` state. - Submitting the form adds a todo to the `todos` array.

2. Render the List

- Todos are rendered using `map()` in a list (`<ul>`, `<li>`).

3. Delete a Todo

- Each todo has a Delete button that filters it from the list by index.

Part 4 – Activity: Add New Features (30 min)

Student Challenges (Pick any):

- Add a checkbox next to each todo to mark it as "Done"
- Style done items with a strikethrough
- Add a way to edit a todo
- Save todos to `localStorage`

Encourage students to research and work in pairs if needed.

Part 5 – Bonus Lab: Component Extraction (15 min)

Student Challenge:

Refactor the code by splitting the Todo item into its own component:

```
// TodoItem.js
function TodoItem({ todo, index, onDelete }) {
  return (
    ⟨li⟩
      {todo}
      ⟨button onClick={() => onDelete(index)}⟩Delete⟨/button⟩
    ⟨/li⟩
  );
}
export default TodoItem;
```

Then in `App.js`, import and use it:

```
import TodoItem from './TodoItem';
// ...
⟨TodoItem todo={todo} index={index} onDelete={handleDeleteTodo} /⟩
```

Part 6 – Review and Wrap-Up (15 min)

Review key concepts from the lab:

- JSX syntax and functional components
- useState and managing dynamic data
- React's declarative rendering approach
- Importance of reusable components

Optional Homework

- Add edit functionality using a second input field
- Read about `useEffect` in the React docs
- Try deploying the app using Netlify or Vercel

Resources

- React Official Docs
- W3Schools React
- Scrimba React Course
- Codecademy React Track