Day 16 – [10th July 2025]

TOPICS COVERED

Reading Multiple Input Fields using One useState:

Instead of using multiple state variables, we can manage multiple input fields using one object with a single useState.

Example:

```
import { useState } from "react";

function Form() {

  const [formData, setFormData] = useState({

    name: "",

    email: ""

  });

  function handleChange(e) {

    setFormData({

      ...formData,

      [e.target.name]: e.target.value

    });

  }

  function handleSubmit(e) {

    e.preventDefault();

    alert(`Name: ${formData.name}, Email: ${formData.email}`);

  }

  return (

    <form onSubmit={handleSubmit}>

      <input name="name" onChange={handleChange} value={formData.name} placeholder="Name"
/>

      <input name="email" onChange={handleChange} value={formData.email} placeholder="Email"
/>

      <button type="submit">Submit</button>

    </form>

  );
}
```

State Uplifting:

State lifting is the process of moving state up to the closest common ancestor when two or more components need to share the same data.

Example:

Child component sends input data to parent using a callback (props).

Parent holds the state.

```
function Child({ onInputChange }) {
  return <input onChange={(e) => onInputChange(e.target.value)} />;
}
function Parent() {
  const [text, setText] = useState("");
  return (
    <div>
      <Child onInputChange={setText} />
      <p>Input from child: {text}</p>
    </div>
  );
}
```

Props Drilling & Its Problem:

Props Drilling happens when you pass data through many levels of components even though only a deep child needs it.

Problem:

- Difficult to maintain and understand
- Unnecessary props passed through intermediate components
- Context API (Solution to Props Drilling):
- React’s Context API allows you to share data globally without passing props manually at every level.

Example:

```
import { createContext, useContext } from "react";
const UserContext = createContext();
```

```
function Parent() {
  return (
    <UserContext.Provider value="Navpreet">
      <Child />
    </UserContext.Provider>
  );
}
function Child() {
  const user = useContext(UserContext);
  return <h1>Hello, {user}</h1>;
}
```

useContext() makes it easy to read shared state in any component.

TOOLS USED:

VS Code

React Dev Server

React Context API

Browser (for testing)