AeroAspire - SDE Intern

Gokul Krishna S

Week 2 – Day 3 (October 1)

Questions/Reflections :

1. Walk through flow: user types → state updates → component re-renders → effect runs (if any).

    - User types in a form (like an input): the onChange handler fires.
    - State updates: The handler calls setState to update a state variable (e.g., setValue(e.target.value)).
    - Component re-renders: React re-runs the component with the new state value, updating the UI.
    - Effects run (if any): After re-render, any useEffect hooks whose dependencies changed will execute. This is where side effects (like fetching data or updating the DOM) happen.

2. How useEffect works: dependencies, cleanup, initial render.

    - Dependencies: The second argument to useEffect, an array, tells React when to run the effect. If the array includes [value], the effect runs whenever value changes.
    - Cleanup: If the effect returns a function, React runs it before running the effect again (on dependency change) or when the component unmounts. Use this to clean up subscriptions, timers, etc.
    - Initial render: If the dependency array is empty ([]), the effect runs only once, just after the first render.

3. What pitfalls exist (e.g. stale closures, infinite loops)?

    - Stale closures: If you forget to list dependencies, effects can use outdated variables. Always list everything you use from outside the effect in the dependency array.

- Infinite loops: If you update state inside an effect without careful dependencies, it can cause endless re-render/effect cycles. Write effects to change state only when necessary, and check dependencies.

4. What is a controlled vs uncontrolled component?

    - Controlled component: React controls the value. The input's value is set by React through value and updated via onChange (e.g., <input value={value} onChange={handleChange} />). State always owns the value.
    - Uncontrolled component: The DOM manages its own state. You use defaultValue and might access the value with refs, not state. Less React control, more manual work.

5. Describe event handling in forms (onChange, onSubmit).

    - onChange: Fires whenever the user types, selects, or otherwise changes the value in a field. You use it to update React state.
    - onSubmit: Fires when a form is submitted. Common practice: call event.preventDefault() to prevent the page reload, then read form state and act (e.g., send to server).

6. How MUI (Material-UI) Helps

    - Theming: MUI lets you define a theme (colors, fonts, spacing) once that applies consistently to all components. This keeps your UI unified.
    - Form helpers: MUI offers ready-made form components (like TextField, FormControl, Select) with accessibility and styling built in.
    - Error display: MUI components support error states (via error and helperText props) that make it easy to show users what's wrong (like invalid input) without custom code.