AeroAspire - SDE Intern

John Nikhil G

Week 2 – Day 3 (October 1)

Questions/Reflections :

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

- When the user enters text into an input, the onChange event triggers.
- The event handler then updates state by calling setState with the new value.
- Because state changes, React re-executes the component function, giving the UI fresh data.
- After the render finishes, any useEffect hooks with updated dependencies will be executed, handling side effects like API calls or DOM updates.

---

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

- **Dependencies:** The array passed as the second argument to useEffect decides when the effect should run. If [count] is provided, it re-runs every time count changes.
- **Initial render:** With an empty dependency array ([]), the effect is triggered only once, right after the very first render.
- **Cleanup:** If you return a function from inside useEffect, React uses it to clean up before re-running the effect or when the component is removed from the DOM. Useful for stopping timers or unsubscribing from listeners.

---

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

- **Infinite loops:** If an effect continuously sets state on every run, and that state is part of its dependencies, the cycle never stops. This locks the app into constant re-renders.
- **Stale closures:** If you don't correctly include all variables in the dependency array, the effect might use "old" versions of data. Always keep the dependency array accurate.

---

4. What is a controlled vs uncontrolled component?

- **Controlled:** The component's form value is tied directly to React state. You provide the value and an onChange that updates it.

React is always the single source of truth for the input.

- **Uncontrolled:** The form element maintains its own internal value, with React not syncing it on every change. You typically rely on defaultValue and refs to read values only when you need them.

---

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

- **onChange:** Runs every time the field's content or selection changes. You usually update state here so React stays in sync with what the user typed.
- **onSubmit:** Fires when the user submits the form. Normally you call event.preventDefault() to stop the default page reload, then use the form's state values to process the submission (like saving or sending data).

---

6. How MUI (Material-UI) Helps

- **Theming:** With MUI you can create a consistent design system (colors, typography, spacing) and apply it across your entire app easily.
- **Form components:** It provides pre-built, styled input elements such as TextField, Select, and FormControl that save time and ensure accessibility.
- **Validation states:** Inputs support built-in error display through props like error and helperText, letting you show validation messages without custom styling.