**React**

**Day 80th**

**useReducer Hook**

**Rules of Hooks**

Hooks area vailable in only functional component. It is a JavaScript function. You need to follow two rules to use hook.

1. Only call hooks at the top level.
2. Don’t call hooks inside loops, conditions, or nested functions.
3. Don’t call hooks from regular JavaScript functions instead you can call :
   1. Call hooks from the react functions component
   2. Call hooks from custom hooks
4. Create or use hooks with use prefix

**Here I will be having scenario and explanation**

**❌** hooks can only be called inside the body of a function component

Scenarios:

1. You might be breaking the rules of hooks
2. You might have mismatching versions of React and React DOM
3. You might have more than one copy of react in the same loop

**Breaking rules of Hooks**

Functions whose names start with use are called ***Hooks*** in react.

Don’t call hooks inside loops, conditions or nested functions. Instead, always use hooks at the top level of your React function, before any early returns. You can only call hooks while React is rendering a function component.

* Call them at the top level in the body of a function component
* Call them at the top level in the body of a custom hook.

It’s not supported to call Hooks (functions starting with use) in any other cases, for example

* Do not call hooks inside conditions or loops
* Do not call Hooks after a conditional return statement
* Do not call hooks in event handlers
* Do not call hooks in class based component
* Do not call hooks inside functions passed to useMemo, useReducer or useEffect

**Mismatching Versions of React and React DOM**

You might be using a version of react-dom (<16.8.0) or react-native (<0.59) that does not yet support hooks. You can run npm ls react-dom or npm ls react-native in your application folder to check which version you’re using. If you find more than one of them, this might also create problems.