**React**

**Day 81st**

**useReducer hook (sir)**

It is similar to `**useState**` hook. If it is complex state data means huge data then use state is not enough. That’s why we need use reducer it helps us to store. The `**useReducer**` is the alternative to the useState hook whenever have complex data. It helps you to manage complex state logic.

ContextAPI with useReducer is the best option to create complex state logic in react application.

For state management like redux, Recoil, Mobx useReducer is the best option compared to useState.

**useRducer (google)**

The useReducer Hook is used to store and update states, just like the useState Hook. It accepts a reducer function as its first parameter and the initial state as the second. useReducer returns an array that holds the current state value and a dispatch function to which you can pass an action and later invoke it.

It allows for custom state logic. If you find yourself keeping track of multiple pieces of state that rely on complex logic, useReducer may be useful.

**useReducer (free code camp)**

Let’s take an example of todo App. This app involves adding, deleting and updating items in the todo list. The update operation itself may involve updating the item or making it as complete.

When you implement a todo list, you’ll have a state variable todoList and make state updates to perform each operation. However, these state updates may appear at different places, sometimes not even inside the component.

To make your code more readable, you can move all your state updates into a single function that can exist outside your component. While performing the required operations, your component just has to call a single method and select the operation it wants to perform.

The function which contains all your state updates is called reducer. This is because you are reducing the state logic into a separate function. The method you call to perform the operations is the dispatch method.

**Usage:**

useReducer is usually preferable to useState when you have complex state logic that involves multiple sub-values or when the next state depends on the previous one.

useReducer also lets you optimize performance for components that trigger deep updates because you can pass dispatch down instead of callbacks.



