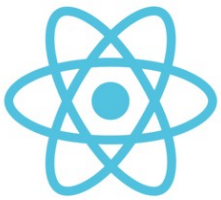

useReducer() React Hooks

- The **useReducer** Hook is similar to the **useState** Hook.
- **useReducer** is a built-in hook that provides an alternative way to **manage state** in functional components.
- It is particularly useful when dealing with **complex state logic** or when the state transitions **depend on previous states**. The useReducer hook follows the **reducer pattern,** similar to how Redux manages state using reducers.
- **Syntax :**

```
useReducer(<reducer>, <initialState>)
```

- **reducer function :** contains your custom state logic. A function that takes the current state and an action as arguments and returns the new state.
- **initialState :** can be a simple value but generally will contain an object. when the component is mounted for the first time.
- **It Returns :**

```
const [state, dispatch] = useReducer()
```

- The useReducer Hook returns the current **state** and a **dispatch** method.
- **state:** The current state value returned by useReducer
- **dispatch**: A function that allows you to send actions to the reducer. It takes an action object as an argument and triggers the state update.
- When you call dispatch, you pass an action object that describes the type of action to be performed along with any additional data needed to update the state.
- The reducer then processes the action and returns a new state based on the current state and the action.

```jsx
import React, { useReducer } from 'react'

const init = {count : 0};

const reducer = (state, action) => {
     switch(action.type){
          case 'INCREMENT':
               return {count: state.count+1};
          case 'DECREMENT':
               return {count: state.count-1};
          default:
               return state;
     }
}
function Usereducer() {

     const [state, dispatch]=useReducer(reducer, init);

     return (
     <>
          <center>
               <h1>Count : {state.count}</h1>
               <button onClick={()=>{dispatch({type:'INCREMENT'})}}>Increment</button>
               <button onClick={()=>{dispatch({type:'DECREMENT'})}}>Decrement</button>
          </center>
     </>
     )
}
export default Usereducer
```

- When the buttons are clicked, the dispatch function is called with the respective action objects, and the reducer function updates the state accordingly.
- useReducer provides a powerful way to manage more complex state logic in React applications and can be used as an alternative to useState when the state management becomes more involved and needs to follow a more structured pattern.
- **Why use Reducer?**
    - Complex state management with multiple variables and actions.
    - State transitions depend on previous states.
    - Centralized and predictable state updates using the reducer pattern.
    - Avoid prop drilling by dispatching actions from any component level.
    - Performance optimization with memoized reducers.
    - Handling middleware and side effects.
    - Better testability due to pure reducer functions.