Day-6

* What are the rules of hooks ?

(1) Only call hooks from a React component function

(2) Only call hooks at the top level.

(3) Can call multiple state or effect hooks inside a component

(4) Make multiple hook calls in the same sequence.

(5) Must call hook before a return statement outside of loops, conditions or nested functions.

* Data fetching using hooks

```
useEffect (() => {
    fetch ('https://.....')
        .then ((response) => response.json())
        .then ((data) => setData (data))
        .catch((error) => console.log (error))
}, [])
```

setData is defined above in a useState hook. So, all the data from API will be stored in that.

Pg.No :
Date: / /20

✶ useReducer hook

→ It is a superpower useState.

→ useState : initialState
useReducer : initialState & reducer function
action object
(has multiple type values)

→ The useReducer hook is best used for more complex data, specifically, arrays and objects.

→ useReducer hook is used to control state of large state-holding objects.

→ e.g.

```
import...

const reducer = (state, action) => {
    if (action.type === "buy") {
        return {money: state.money - 10}
    }

    if (action.type === "sell") {
        return {money: state.money + 10}
    }

    return state
}
```

Pg.No :
Date: / /20

```
function App() {

  const initialState = { money: 100 }
  const [state, dispatch] = useReducer
                  (reducer, initialState)

  return (
   <><h1> {state.money} </h1>

     <div>
       <button onClick={() => dispatch(
           {type: 'buy'})} >
         Buy
       </button>

      <button onClick={() => dispatch ({type: 'sell'})}
           Sell
      </button>
     </div>
  </>
```

* useRef hook

→ Allows to directly create a reference to the DOM element in the functional component.

→ If we change a value in useRef it will not re-render the component.

→ Returns an object