# Lecture 07: useEffect Hook & React.memo in React



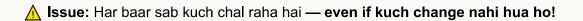
"First Think Why, Then Code How!"

## Problem Shuru Hone Se Pehle (Before useEffect)

React mein har render pe pura component function dubara run hota hai — chahe zarurat ho ya na ho 😩

Ismein include hota hai:

- DOM update
- API call
- Console logging
- Event listeners





#### 😕 So, What's the Need?

#### React se kehna hai:

"Bhai, ye logic sirf tab chalao jab kisi particular cheez mein change aaye."



That's exactly what useEffect karta hai!

## What is useEffect?

useEffect() ek React Hook hai jo allow karta hai:

- Side Effects ko chalana
- Dependencies ke according control karna
- (S) Right time pe run karna (after rendering)

## 3 Types of useEffect:

#### ♦ 1 With Dependency Array [color]

```
useEffect(() => {
  document.body.style.backgroundColor = color;
}, [color]);
```

- Ye code sirf tab chalega jab color change hoga
- ★ Real Use Case:
  - Theme ya background color update
  - Search input change pe fetch karna

## ♦ 2 With Empty Array []

```
useEffect(() => {
  console.log("Component Mounted");
}, []);
```

Ye code sirf ek baar chalta hai — jab component first time render hota hai.

#### Perfect for:

- API calls
- Event listener setup
- Local storage access
- Same as: componentDidMount() in class components

#### ♦ 3 Without Any Array 💢

```
useEffect(() => {
  console.log("Har render pe chalega");
});
```

⚠ Ye code har render ke baad chalega — chahe kuch change hua ho ya nahi.

- Mostly for:
  - Debugging
  - Special animation triggers
    - Avoid this in real apps unless required

## Real Project: Background Color Changer

#### **♀** Goal:

Click karte hi background color change ho jaaye — lekin unnecessary re-renders avoid ho.

#### Problem Without Optimization

- Jab count change hota hai (parent update),
- Tab Colorful (child) bina wajah re-render hota hai
- Even though prop "harshal" same hi hai
  - Waste of time, memory, performance...

#### React.memo Ka Jadoo

```
export default React.memo(Colorful);
```

- React.memo ek HOC (Higher Order Component) hai jo:
  - Props change nahi hote → Re-render nahi hoga

Props change hote hai → Re-render hoga

Simple: "Agar aap same data de rahe ho, to main dobara kaam nahi karunga."

# **Second Second S**

#### index.html

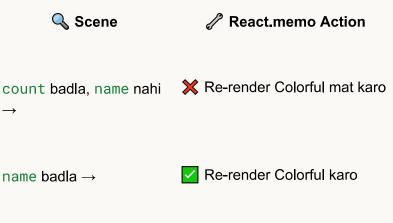
#### script.js (Main Component)

```
function Main() {
 const [count, setCount] = useState(0); // Counter state
 return (
   <>
     <div className="counter">
       <h1>Count is: {count}</h1>
       <button
         onClick={() => setCount(count + 1)}
        style={{ backgroundColor: "white", color: "aqua" }}
         Increment
       </button>
     </div>
     {/* • Pass constant prop to avoid unnecessary render */}
     <Colorful name="harshal" />
   </>
 );
}
// @ Render the app
ReactDOM.createRoot(document.getElementById("root")).render(<Main />);
```

#### bgcolor.js (Child Component)

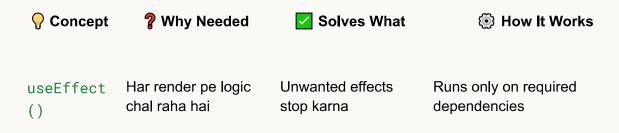
```
import React, { useEffect, useState } from "react";
function Colorful({ name }) {
 const [color, setColor] = useState("black"); // Default color
 console.log("@ Colorful Rendered");
 console.log(" Name prop: ", name);
  // ✓ useEffect runs only when color changes
  useEffect(() => {
    console.log(") useEffect executed");
    document.body.style.backgroundColor = color;
  }, [color]);
  return (
    <>
      <h1>
   Background Color Changer</h1>
      <div>
        {/* Duttons for different colors */}
        <button style={{ backgroundColor: "red" }} onClick={() =>
setColor("red")}>Red</button>
        <button style={{ backgroundColor: "blue" }} onClick={() =>
setColor("blue")}>Blue</button>
        <button style={{ backgroundColor: "orange" }} onClick={() =>
setColor("orange")}>Orange</button>
```

## Real-World Thought Process



**@ Result:** Faster app + No useless work!

## Final Summary: First Thought Se Samjho



[value]	Sirf specific value change pe run	Controlled logic	Monitors only selected variable(s)
[]	Sirf ek baar run karna hai	Initial setup karne mein help	Runs once, on mount
No array	Har render pe chahiye (rare)	Debug ya force run	Runs after every render
React.mem o()	Har baar child render na ho	Performance improvement	Remembers component — runs only if needed

## Final Thought:

"React har baar poora component dubara chalata hai — lekin har logic ko har baar chalane ki zarurat nahi hoti."

Q useEffect se aap logic ko sahi waqt pe chala sakte ho



React.memo se unnecessary re-renders se bacha sakte ho



- Jab bhi DOM ya side-effect ka kaam ho → useEffect mein daalo
- Jab parent update se child bar-bar render ho  $\rightarrow$  React.memo use karo
- Har jagah memo mat thoko sirf jahan actual performance benefit ho

Ye the Lecture 07 ke First-Thought Based Hinglish Notes — Easy to revise, full concept clear, aur interview ke liye bhi helpful.