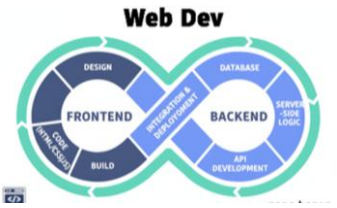
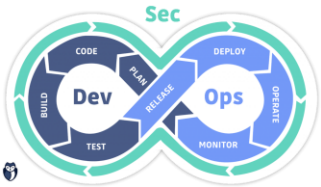



# Day 56

# "Web Development + Security"

## The useEffect Hook in React:

**What Is useEffect?**

The useEffect() Hook in React lets you run side effects in functional components. A side effect is anything that affects something *outside* the component — like:

- Fetching data from an API
- Updating the DOM manually
- Setting up a timer or event listener
- Logging or interacting with browser storage

**Why We Need It**

Before Hooks, you could only handle side effects in class components using lifecycle methods like:

- componentDidMount()
- componentDidUpdate()
- componentWillUnmount()

useEffect() gives functional components that same power.

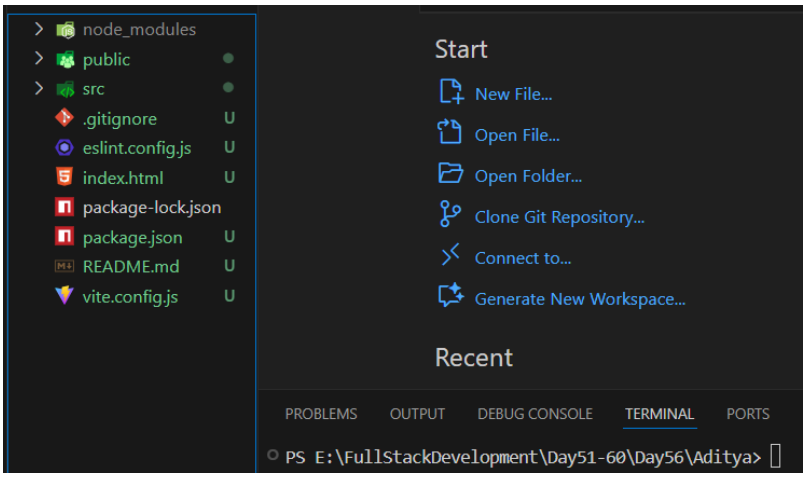
**Syntax**

```
useEffect(() => {
  // side effect code
  return () => {
    // cleanup code (optional)
  };
}, [dependencies]);
```

**Parameters:**

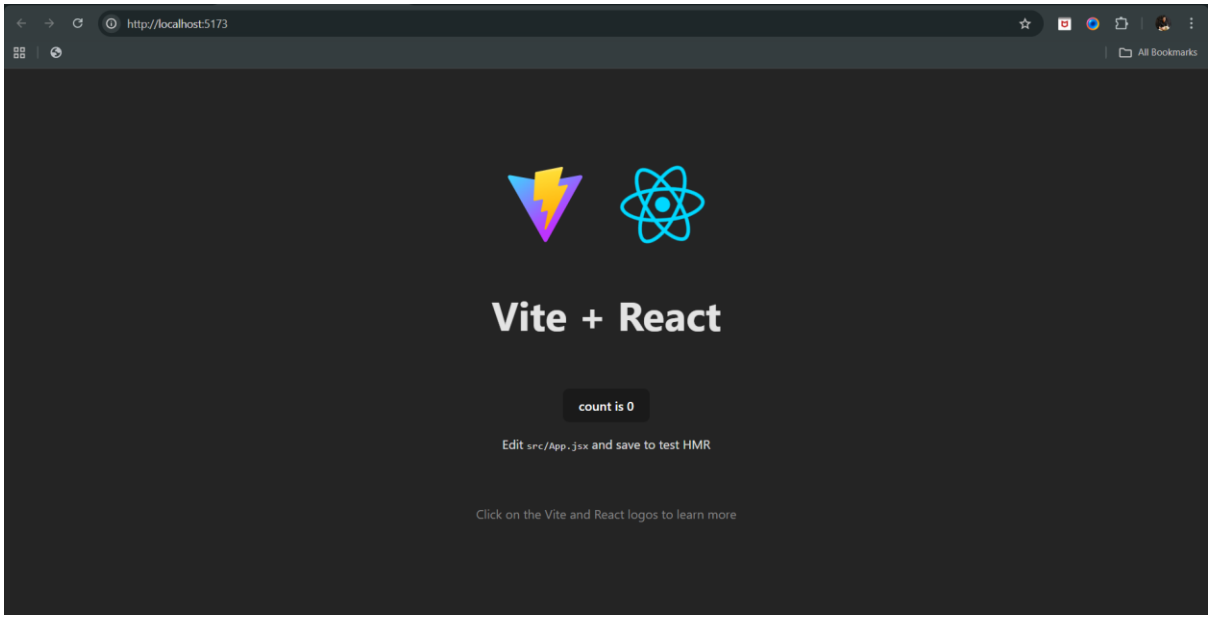
- Callback Function: Code to run after render.
- Dependency Array: Controls when the effect runs.
- Cleanup Function (optional): Cleans up side effects (like removing timers or listeners).

Now, to understand this we will first create the project like earlier, following file directory will come:




Now, just open run code via terminal:

Output:



Now, in order to understand the very basic use case of useEffect then we can, add the alert box to the above page, when ever it loads:

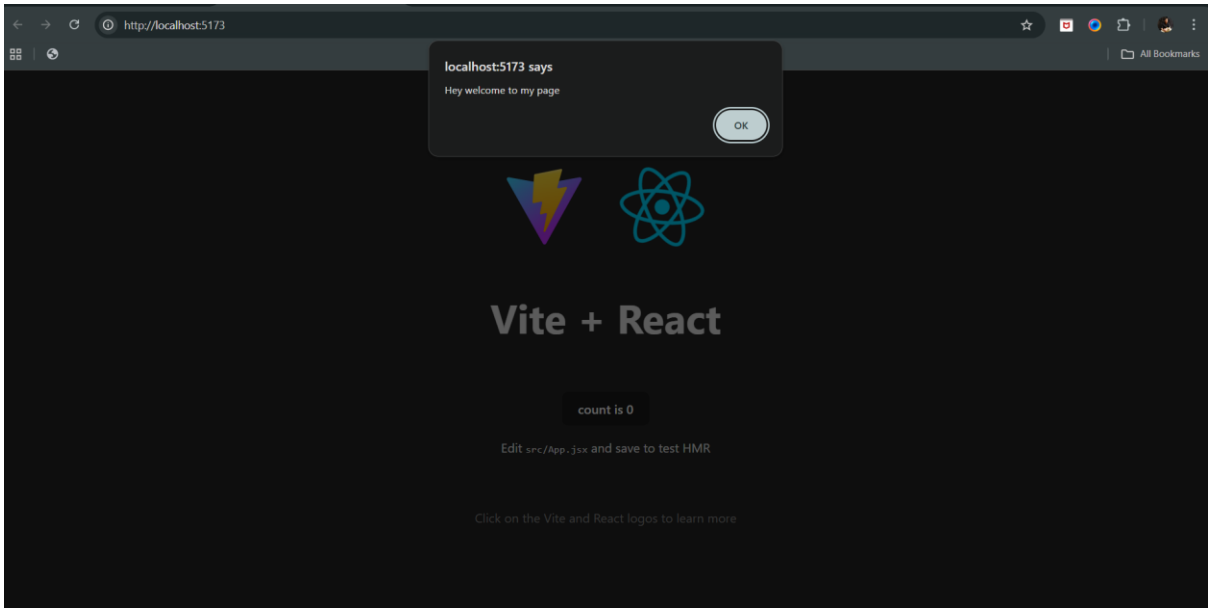
App.jsx: 9 to 11 added for that alert box

App.jsx U

src > App.jsx > App > useEffect() callback

```jsx
import { useEffect, useEffectEvent, useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  useEffect(() => {
    alert("Hey welcome to my page")
  }, [])


  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Vite + React</h1>
      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.jsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
```

Output: so basically the alert box came as a side effect when the actual got rendered.



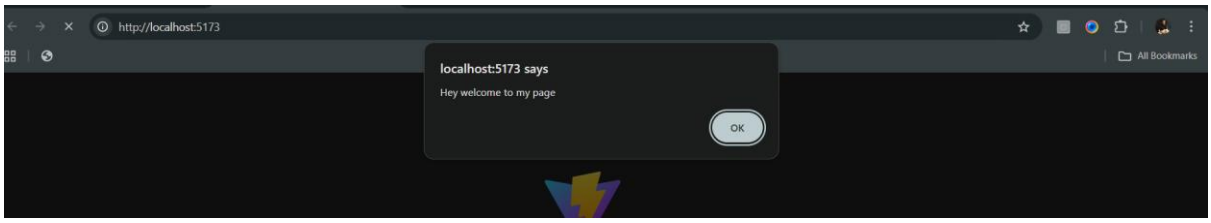
Now, suppose we want that when ever the count get change, the alert box will come (apart from the one which is coming on the loading of website).

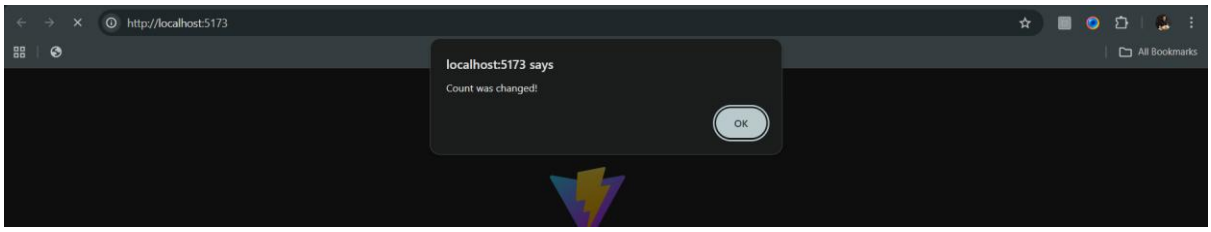
For that we will add one more useEffect hook, and pass the count in the array as the 3rd argument:

App.jsx: snippet of code

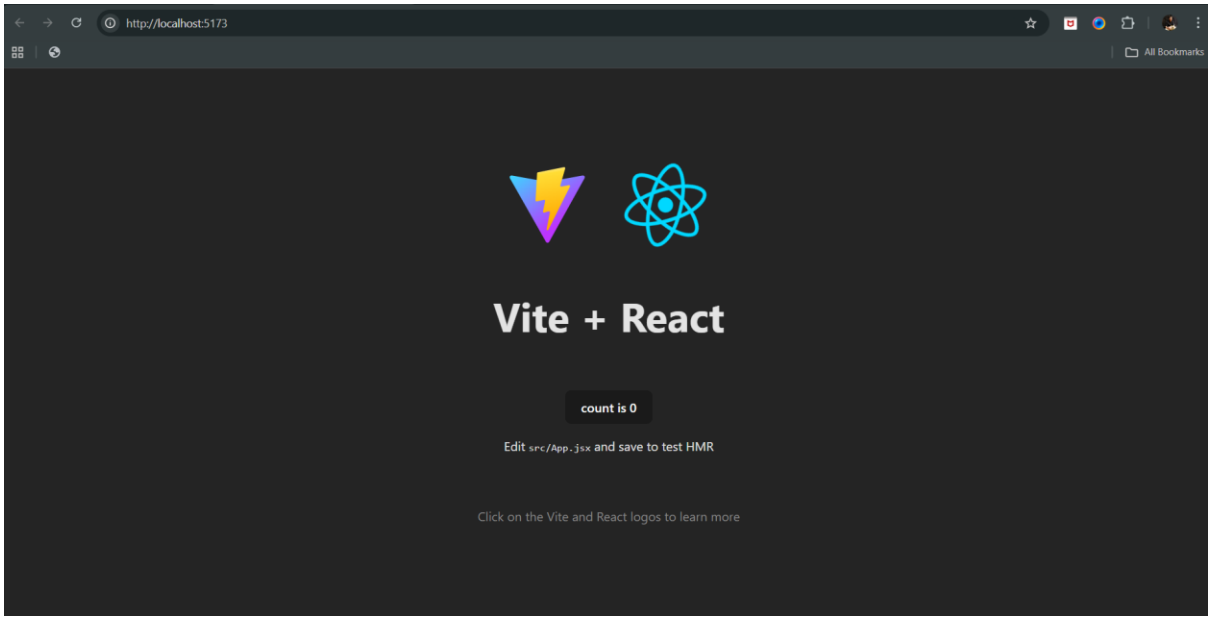
```jsx
import { useEffect, useEffectEvent, useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  useEffect(() => {
    alert("Hey welcome to my page")
  }, [])

  useEffect(() => {
    alert("Count was changed!")
  }, [count])


  return (
```

Output: first alert box with the “Hey welcome to my page” will come and then “Count was changed” another alert box will come (on load 0 count is also taken as count changed).

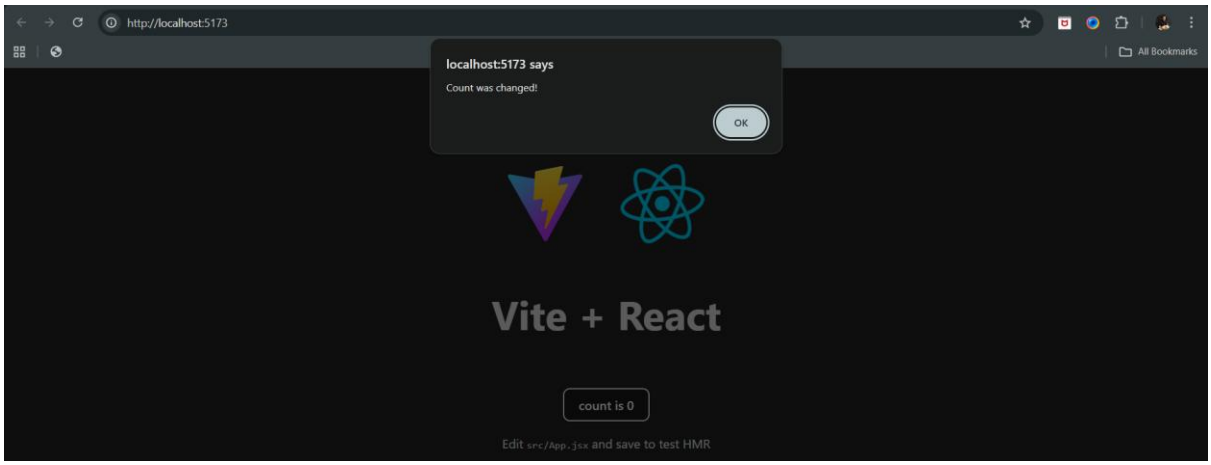


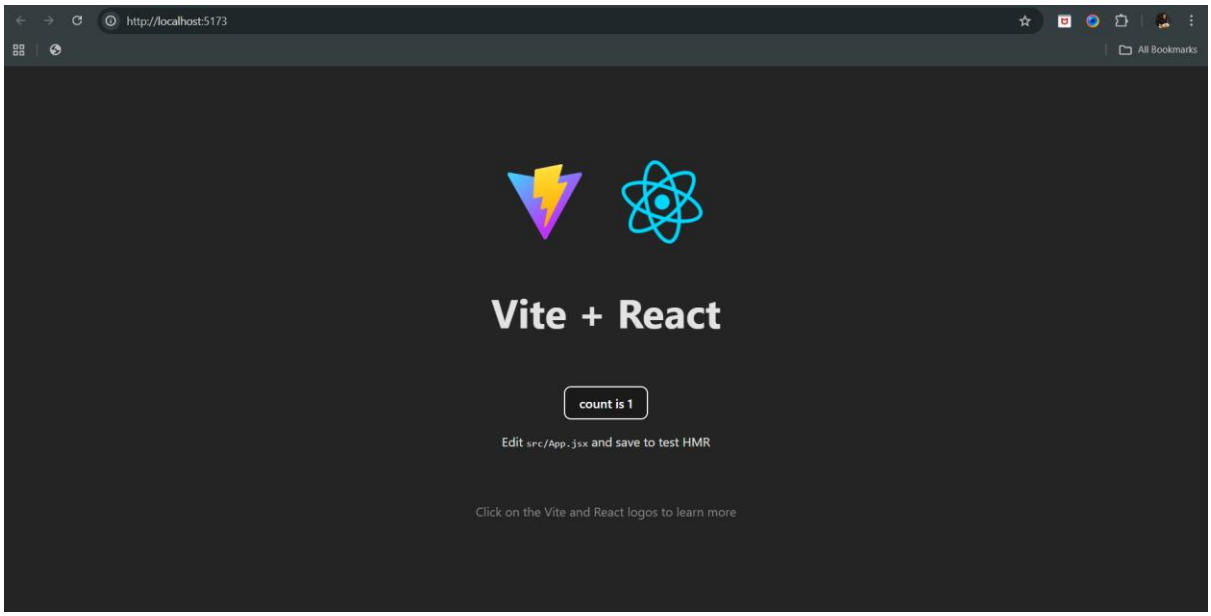



Output: actual output




Then output when count button was clicked:

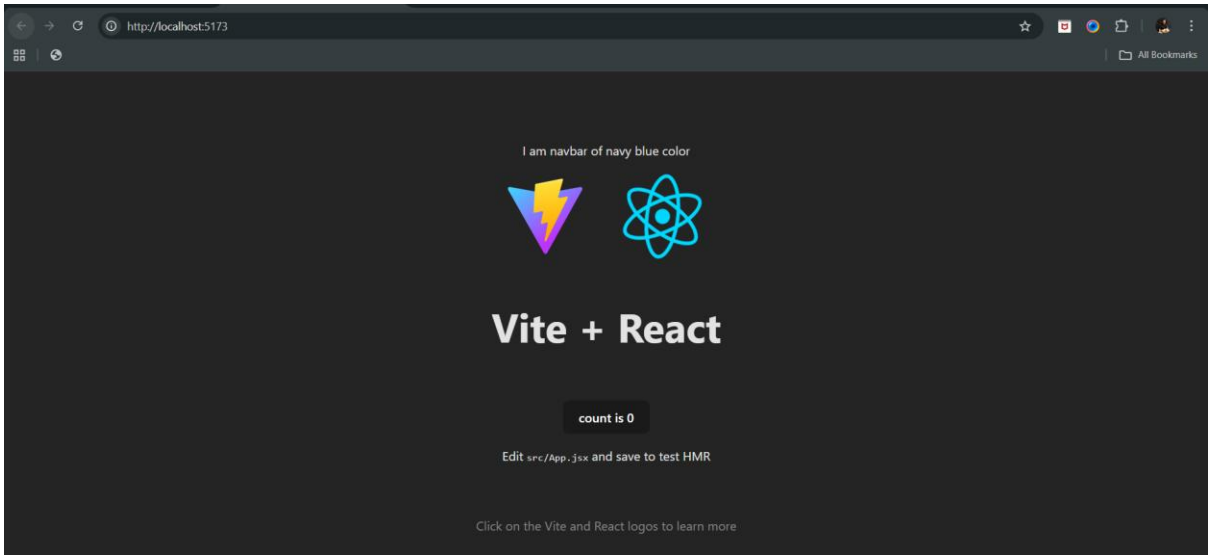


Now, to understand in a better way, we may intend to add the navbar in the page, for which we will create a folder named components, and inside it we will create a Navbar.jsx with the following code:


Then import it to the App.jsx: at line number 5 and 20.


Output: after those 2 alert box, it will come as expected.



Now, to understand in a much better way we will use the useEffect in the Navbar.jsx: this will work whenever the colour will change.

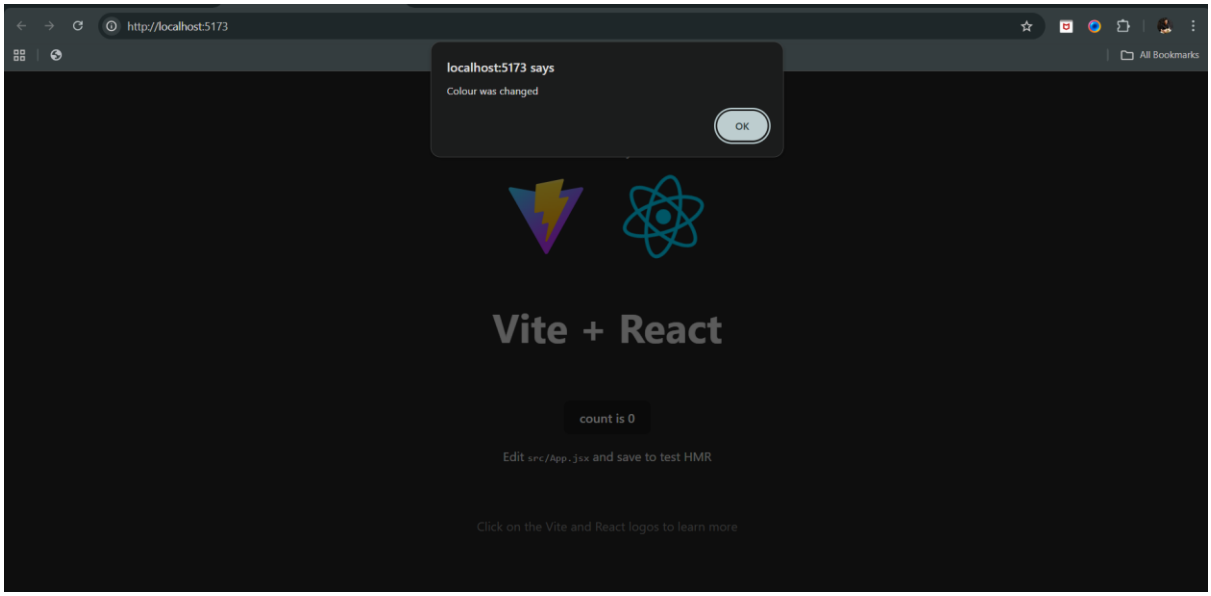
Navbar.jsx:

App.jsx U | Navbar.jsx U | main.jsx U

src > components > Navbar.jsx > ...

```jsx
import React, {useEffect, useEffectEvent} from 'react'

const Navbar = ({color}) => {
    useEffect(() => {
      alert("Colour was changed")
    }, [color])

  return (
    <div>
      I am navbar of {color} color
    </div>
  )
}

export default Navbar
```

Output: since by default first it will consider that colour has changed. Rest will occur as earlier.



Now, can we add a useEffect which will run on every render?  Yes, we can do so, by removing the [] from the useEffect syntax.

```
const Navbar = ({color}) => {
    useEffect(() => {
      alert("Colour was changed")
    })
```

For case we want that useEffect should run on first render:

```
useEffect(() => {
  alert("Colour was changed")
}, [])
```

For case we want that useEffect should run when any value changes:

```
useEffect(() => {
  alert("Colour was changed")
}, [color])
```

--The End--