

**React-এ Hook** হলো এমন একটা বিশেষ ফাংশন, যেটা তোমাকে function component এর ভেতরে React-এর ফিচার (state, lifecycle, context ইত্যাদি) ব্যবহার করার সুযোগ দেয় — কোনো class component না লিখেই।

### Hook-এর কাজ কী?

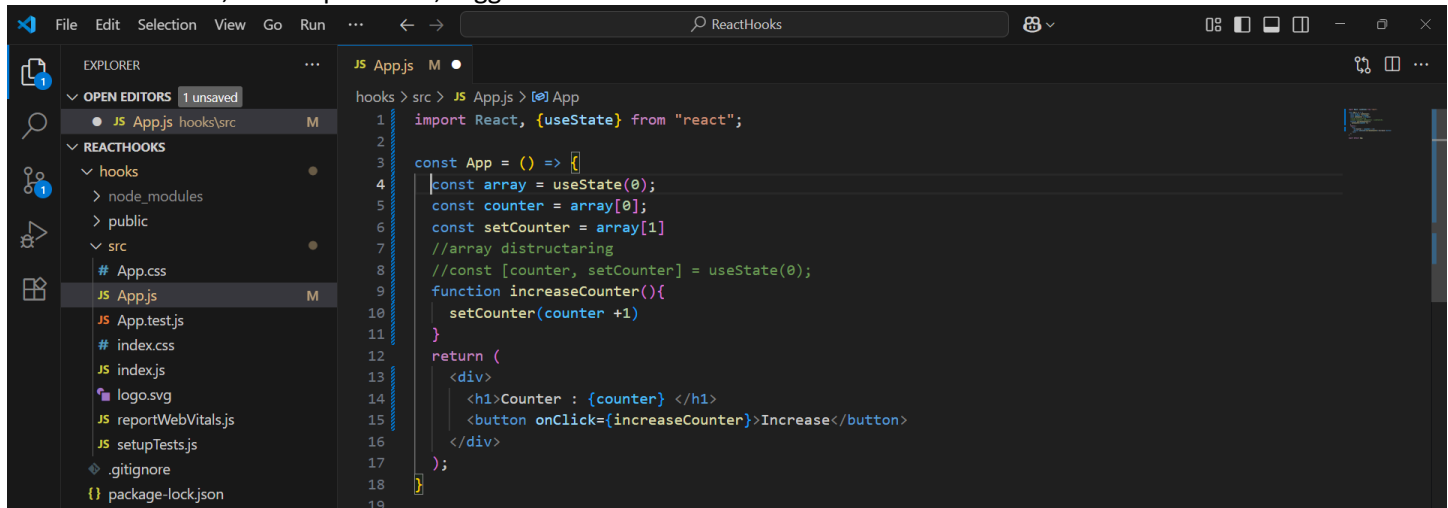
- Component-এর state ম্যানেজ করা (useState, useReducer)
- Render-এর আগে বা পরে side effect চালানো (useEffect, useLayoutEffect)
- Component-এর মধ্যে data শেয়ার করা (useContext)
- DOM element বা persistent value handle করা (useRef)
- Performance optimize করা (useMemo, useCallback)
- নিজের reusable logic তৈরি করা (Custom Hooks)

**Hook = React-এর শক্তি + function component-এর সরলতা**

**1. useState** — State রাখা ও পরিবর্তন করা (Add state in functional component, it is just value or variables of component)

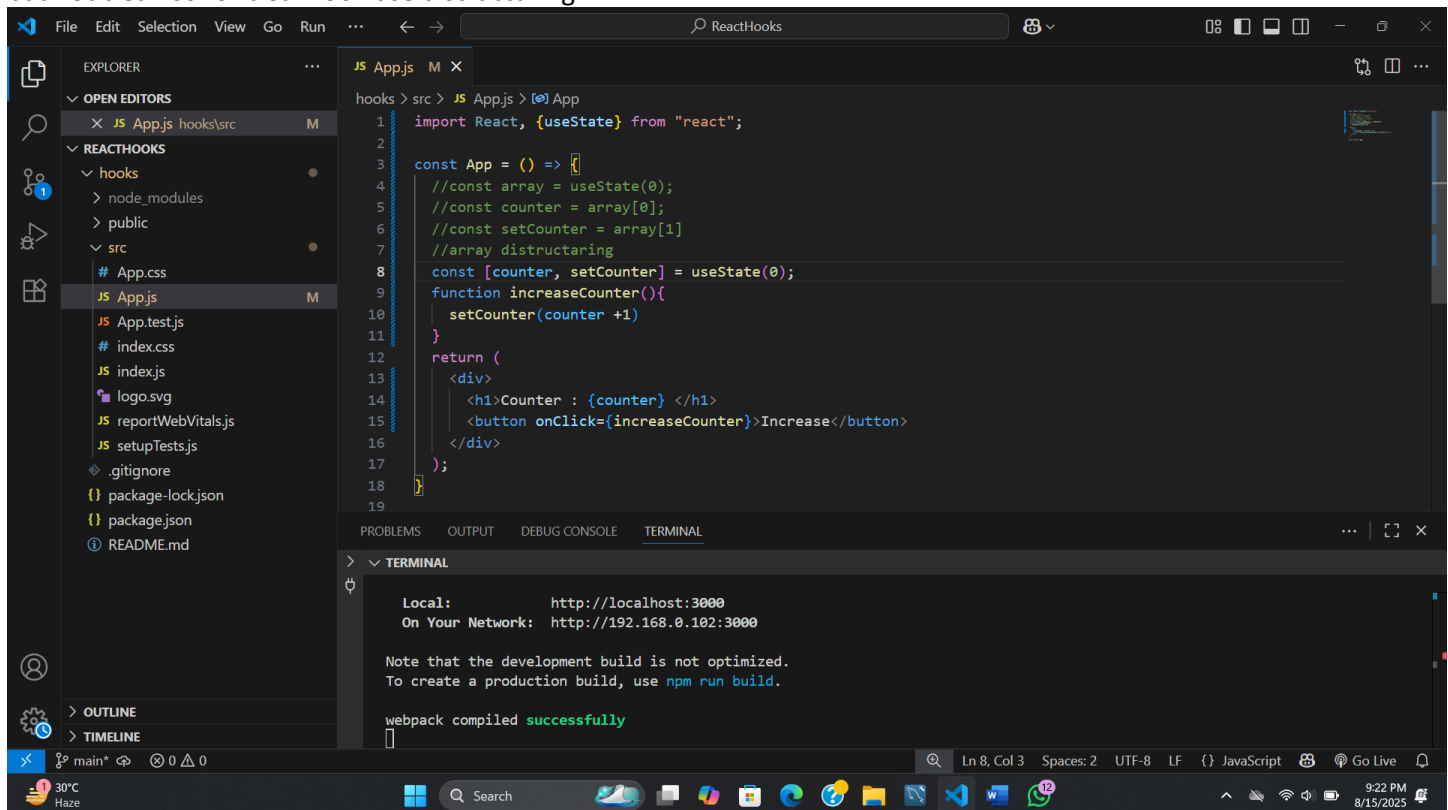
কাজ: Component-এর ভিতরে ডেটা রাখা ও আপডেট করা।

উদাহরণ: Counter, form input value, toggle button state।



```
1 import React, {useState} from "react";
2
3 const App = () => {
4   const array = useState(0);
5   const counter = array[0];
6   const setCounter = array[1]
7   //array destructuring
8   //const [counter, setCounter] = useState(0);
9   function increaseCounter(){
10     setCounter(counter +1)
11   }
12   return (
13     <div>
14       <h1>Counter : {counter} </h1>
15       <button onClick={increaseCounter}>Increase</button>
16     </div>
17   );
18 }
```

but not clean so for clean look use destructuring



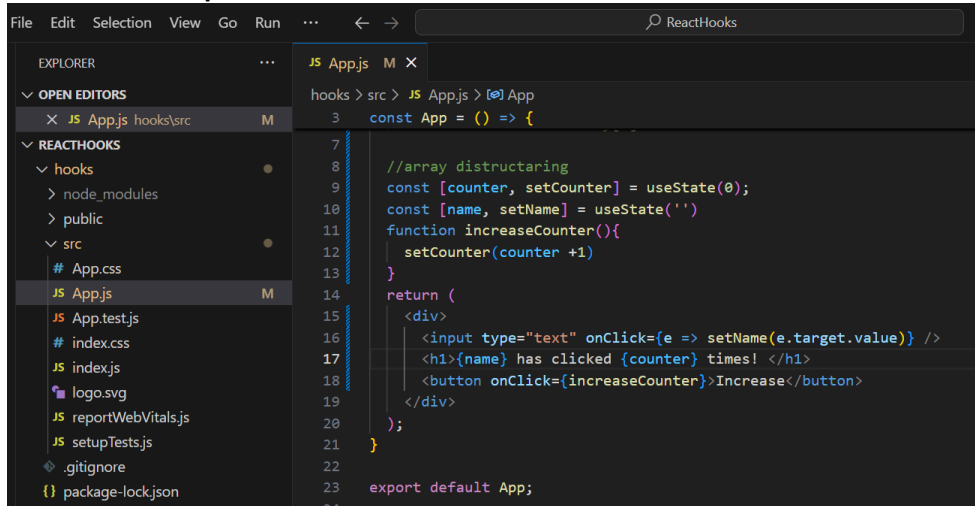
```
1 import React, {useState} from "react";
2
3 const App = () => {
4   //const array = useState(0);
5   //const counter = array[0];
6   //const setCounter = array[1]
7   //array destructuring
8   const [counter, setCounter] = useState(0);
9   function increaseCounter(){
10     setCounter(counter +1)
11   }
12   return (
13     <div>
14       <h1>Counter : {counter} </h1>
15       <button onClick={increaseCounter}>Increase</button>
16     </div>
17   );
18 }
```

Local: http://localhost:3000  
On Your Network: http://192.168.0.102:3000

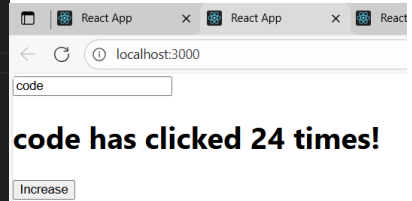
Note that the development build is not optimized.  
To create a production build, use `npm run build`.

webpack compiled successfully

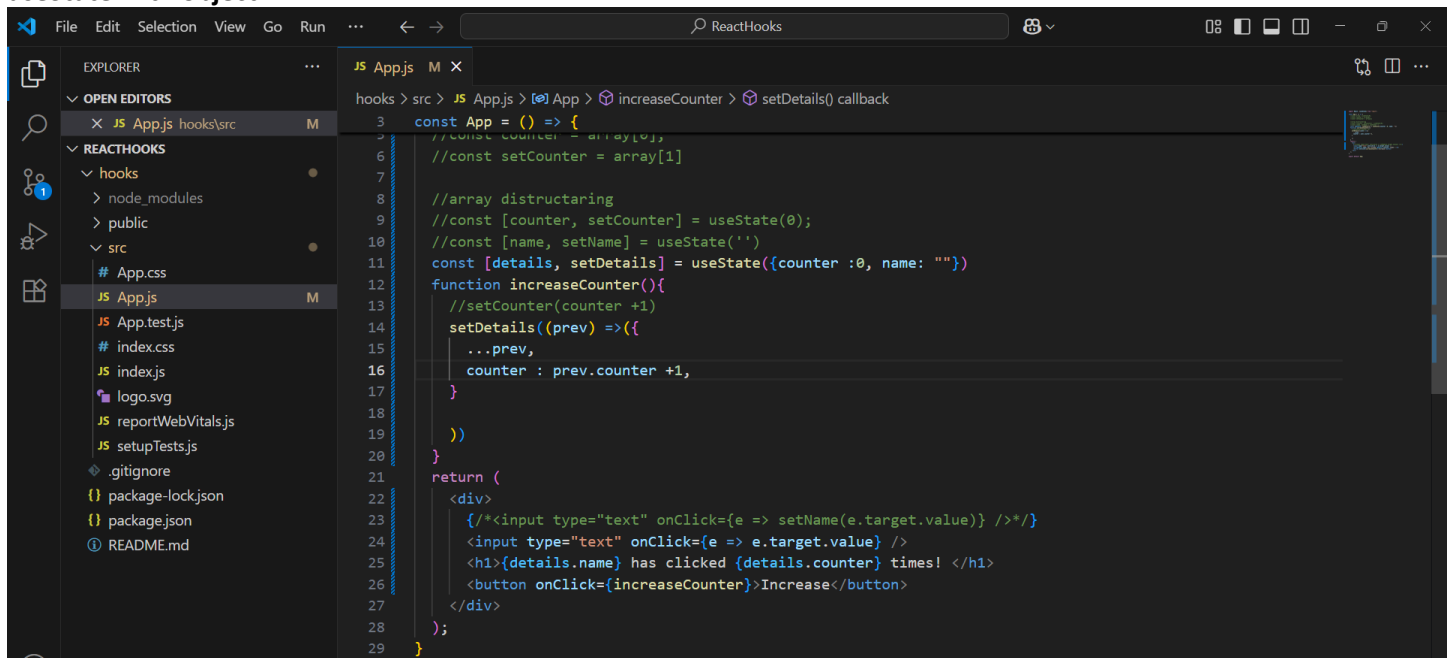
## useState with input text ->



```
3  const App = () => {
7
8    //array destructuring
9    const [counter, setCounter] = useState(0);
10   const [name, setName] = useState('');
11   function increaseCounter(){
12     setCounter(counter + 1)
13   }
14   return (
15     <div>
16       <input type="text" onClick={e => setName(e.target.value)} />
17       <h1>{name} has clicked {counter} times! </h1>
18       <button onClick={increaseCounter}>Increase</button>
19     </div>
20   );
21 }
22
23 export default App;
```



## useState with object ->



```
3  const App = () => {
4    //const counter = array[0],
5    //const setCounter = array[1]
6
7    //array destructuring
8    //const [counter, setCounter] = useState(0);
9    //const [name, setName] = useState('')
10   const [details, setDetails] = useState({counter : 0, name: ""})
11   function increaseCounter(){
12     //setCounter(counter + 1)
13     setDetails((prev) =>({
14       ...prev,
15       counter : prev.counter + 1,
16     }
17   ))
18   }
19   return (
20     <div>
21       { /*<input type="text" onClick={e => setName(e.target.value)} /> */ }
22       <input type="text" onClick={e => e.target.value} />
23       <h1>{details.name} has clicked {details.counter} times! </h1>
24       <button onClick={increaseCounter}>Increase</button>
25     </div>
26   );
27 }
28
29 }
```

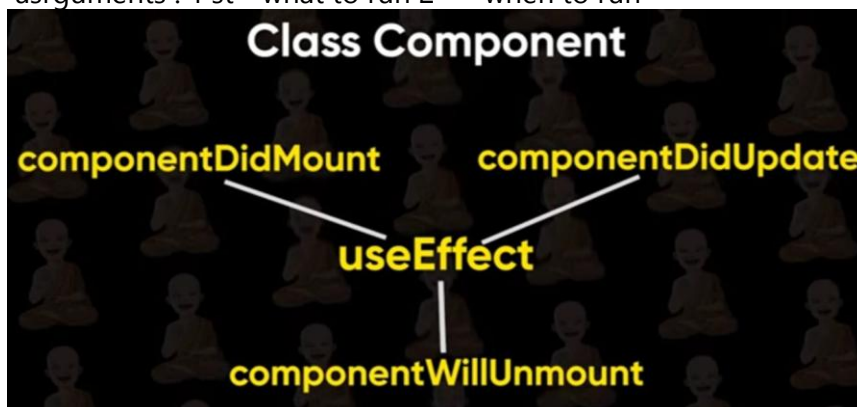
## 2. useEffect — Side Effect চালানো

কাজ: Render হওয়ার পর কাজ চালানো (API call, event listener add/remove, data sync)।

উদাহরণ: Page load-এ ডেটা fetch, window resize listener, updating the DOM document setTimeout and setInterval

**useEffect( callback, dependencies )** dependencies=> array of variables(optional)

arguments : 1<sup>st</sup>= what to run 2<sup>nd</sup>= when to run

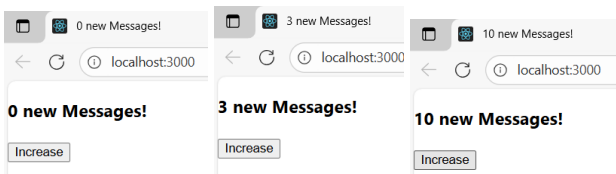


### VARIATION OF USEEFFECT

- useEffect without dependencies
- useEffect with empty array
- useEffect with variables

**useEffect without dependency =>**

useeffect will change in ever single Change in component



```

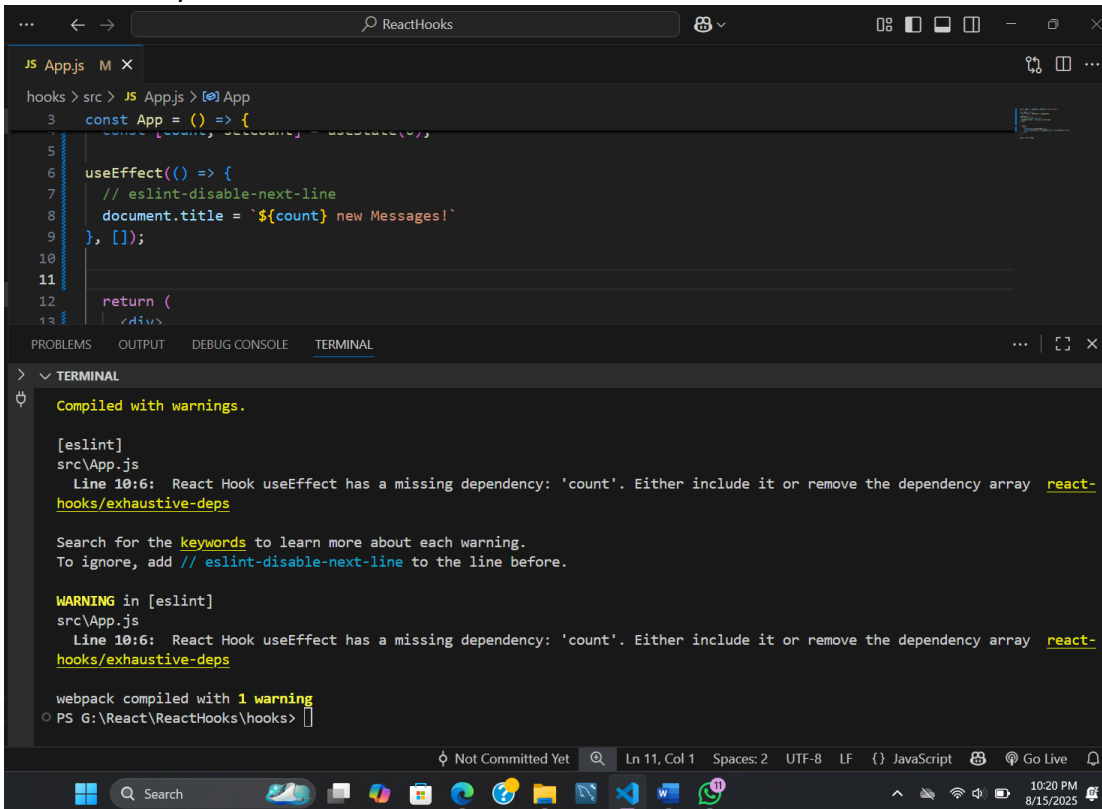
hooks > src > JS App.js > [App] App > useEffect() callback
import React, {useState, useEffect} from "react";

const App = () => {
  const [count, setCount] = useState(0);
  useEffect(() => {
    document.title = `${count} new Messages!`
  })
  return (
    <div>
      <h3>{count} new Messages! </h3>
      <button onClick={() => setCount(count + 1)}>Increase</button>
    </div>
  );
}

export default App;

```

**useEffect with an empty array =>**  
it will run only one time.



**useEffect with variables =>**

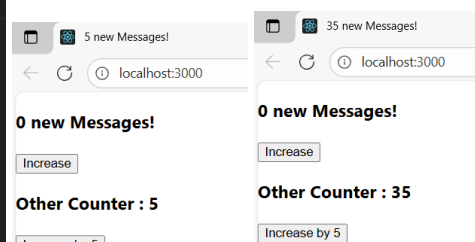
```

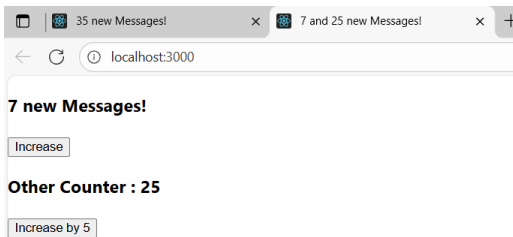
hooks > src > JS App.js > [App] App
import React, {useState, useEffect} from "react";

const App = () => {
  const [count, setCount] = useState(0);
  const [otherCount, setotherCount] = useState(5);
  //useEffect(() => {
  //  // document.title = `${count} new Messages!`
  //})
  useEffect(() => {
    document.title = `${otherCount} new Messages!`;
  }, [otherCount]);
  return (
    <div>
      <h3>{count} new Messages! </h3>
      <button onClick={() => setCount(count + 1)}>Increase</button>
      <h3>Other Counter : {otherCount} </h3>
      <button onClick={() => setotherCount(otherCount + 5)}>Increase by 5</button>
    </div>
  );
}

export default App;

```



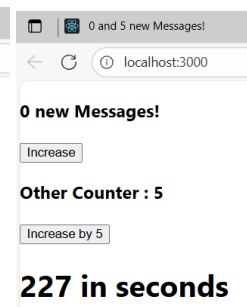
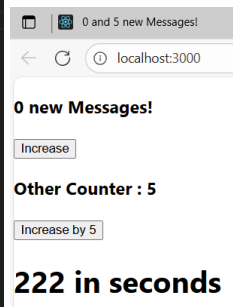
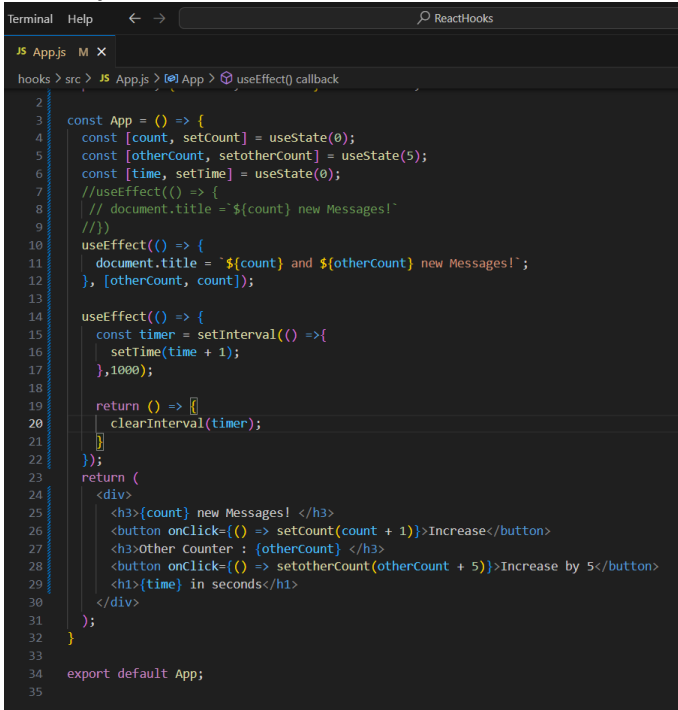


```

9 //})
10 useEffect(() => {    const count: number
11     document.title = `${count} and ${otherCount} new Messages!`;
12 }, [otherCount, count]);
13 return (

```

## Clean-up function in useEffect:



```
import { useState, useEffect } from "react";
```

```
function Counter() {
```

```
  const [count, setCount] = useState(0);
```

```
  useEffect(() => {
```

```
    console.log("Run useEffect", count);
```

```
    return () => {
```

```
      console.log("Clean up", count);
```

```
    };
```

```
  }, [count]);
```

```
  return (
```

```
    <div>
```

```
      <h3>Count: {count}</h3>
```

```
      <button onClick={() => setCount(count + 1)}>Increase</button>
```

```
    </div>
```

```
  );
```

```
}
```

```
export default Counter;
```

### 3 VARIATION OF USEEFFECT

- **useEffect without dependencies** - it runs with first render and also run on any thing changes.
- **useEffect with empty array** - it runs only on first render.
- **useEffect with variables** - it runs on first render and runs with that variable change.