

Internship Report – Frontend Dev

Week 7: React.js Advanced

Name: Zainab

Father Name: Assad Qayyum

Date: 7th Aug, 2025

Internship Domain: Front-end Intern

Task: Custom Hooks (basic intro)

Task Overview: (Day4)

Today's task focused on learning the concept of **Custom Hooks in React**. The goal was to understand how to reuse logic between components, keep code clean, and improve project structure by creating and using our own hooks.

Content Covered:

The content covered is:

- What are React Hooks?
- What are Custom Hooks?
- Why and when to use Custom Hooks
- Rules for creating Custom Hooks

Hooks in React:

Hooks are special functions that let you “hook into” React’s features like state, lifecycle, or context. These were introduced in React 16.8. Examples of built-in hooks:

- `useState` → adds state to function components.
- `useEffect` → performs side effects (like data fetching).
- `useContext` → uses context values.

Custom Hooks (Basic Intro)

A Custom Hook is a function that starts with the word `use` and **uses one or more built-in hooks** like `useState`, `useEffect`, etc., inside it.

A **Custom Hook** is a **JavaScript** function that:

- starts with the word `use`
- can use built-in hooks inside it
- helps you reuse logic across multiple components

Why use Custom Hooks?

- To avoid repeating code in multiple components
- To make components simpler and easier to read
- To separate logic from UI
- To make code reusable and organized

Key Properties of Custom Hooks

Property	Description
Starts with <code>use</code>	Must always start with <code>use</code> , to follow React's rules
Reusable	Can be used in multiple components
Uses other hooks	You can use built-in hooks like <code>useState</code> , <code>useEffect</code> etc
No JSX inside	Custom hooks do not return JSX, only values or functions

Parts of Custom Hook

1. **Function definition:** Should start with `use`, like `useCounter`
2. **Hook logic:** Use `useState`, `useEffect`, etc.
3. **Return:** Return any data or functions needed
4. **Import and use:** Can be used in any component like a normal hook

Rules of Custom Hooks

Start with the word use Example: useForm, useAuth, useTimer

Can use other hooks like useState, useEffect inside

Follow the Rules of Hooks:

- Only call hooks at the top level
- Only call hooks inside React functions or other custom hooks

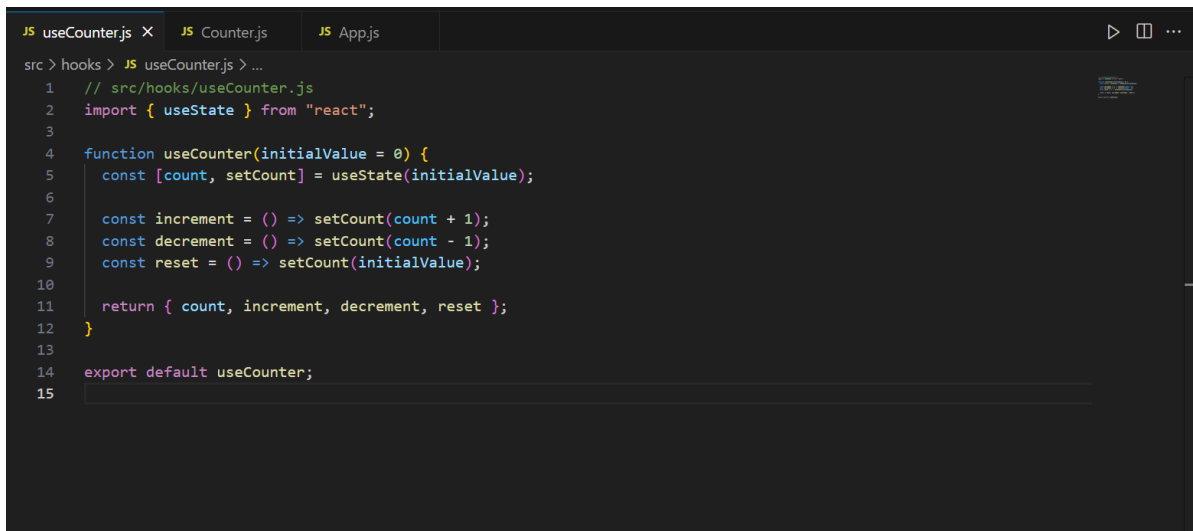
Structure of custom hook:

```
function useSomething() {  
  // useState  
  // useEffect  
  // your logic here  
  
  return something; // return what the hook provides  
}
```

Practice Code:

- This Code uses a **custom hook useCounter** to manage a **counter's state** with increment, decrement, and reset functions.
- The **Counter component** displays the **current count and buttons to control it**. Styling is applied using inline CSS.
- The custom hook keeps logic reusable and separates from UI components.

useCounter.js:



```
JS useCounter.js X JS Counter.js JS App.js  
src > hooks > JS useCounter.js > ...  
1 // src/hooks/useCounter.js  
2 import { useState } from "react";  
3  
4 function useCounter(initialValue = 0) {  
5   const [count, setCount] = useState(initialValue);  
6  
7   const increment = () => setCount(count + 1);  
8   const decrement = () => setCount(count - 1);  
9   const reset = () => setCount(initialValue);  
10  
11   return { count, increment, decrement, reset };  
12 }  
13  
14 export default useCounter;  
15
```

Counter.js:

```
JS useCounter.js  JS Counter.js X  JS App.js
src > components > JS Counter.js > ...
1  import useCounter from "../hooks/useCounter";
2
3  function Counter() {
4    const { count, increment, decrement, reset } = useCounter(0);
5
6    return (
7      <div style={{
8        textAlign: "center",
9        marginTop: "50px",
10       fontFamily: "Arial, sans-serif"
11      }}>
12        <h1 style={{
13          color: "#000",
14          fontWeight: "bold",
15          marginBottom: "10px",
16          fontSize: "36px" // Larger font size
17        }}>
18          My React App
19        </h1>
20        <h2 style={{
21          color: "#000",
22          fontWeight: "bold",
23          marginBottom: "30px",
24          fontSize: "28px" // Larger font size
25        }}>
26          Count: {count}
27        </h2>
28
29        <div>
30          <button
```

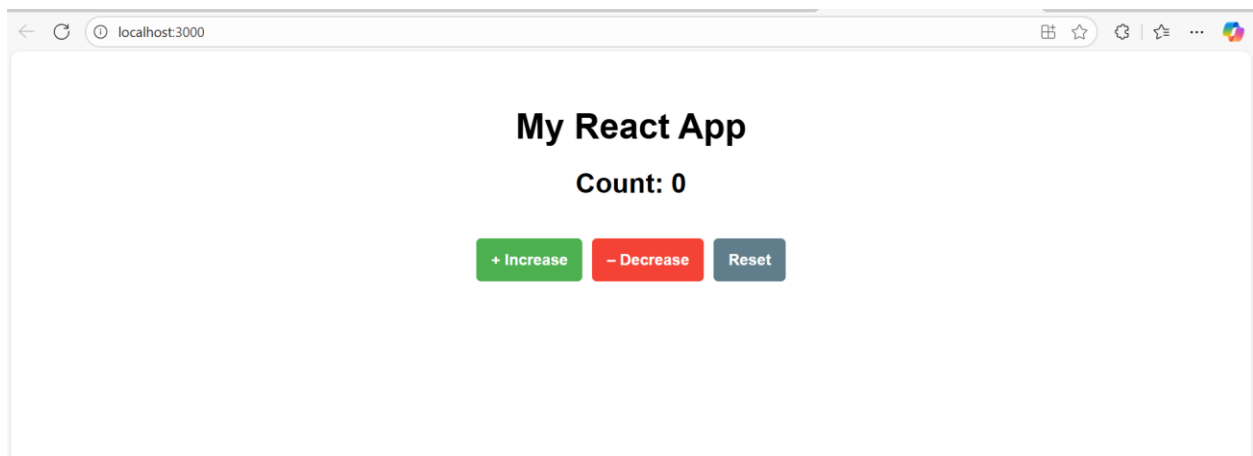
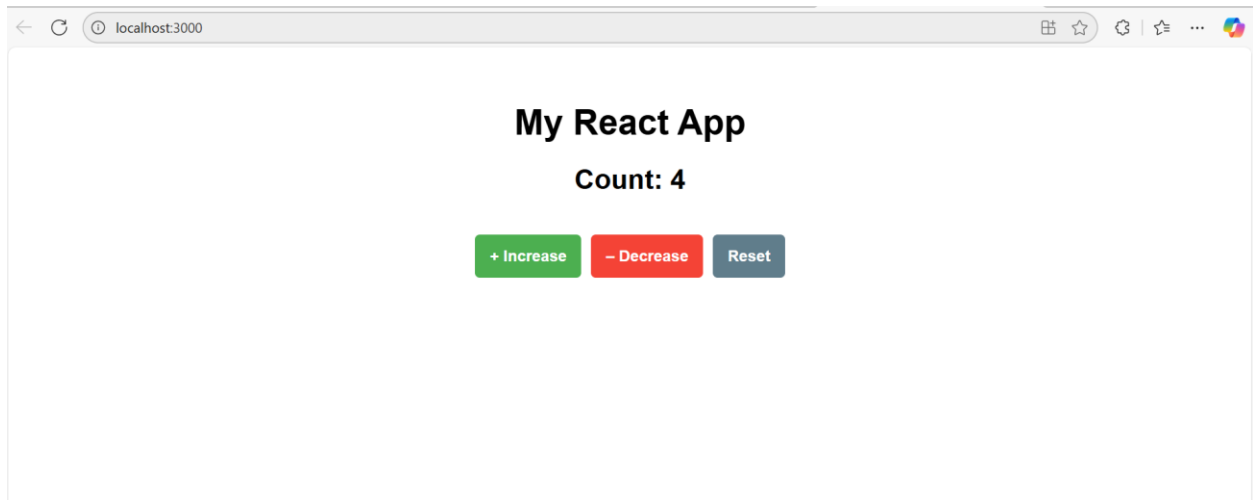
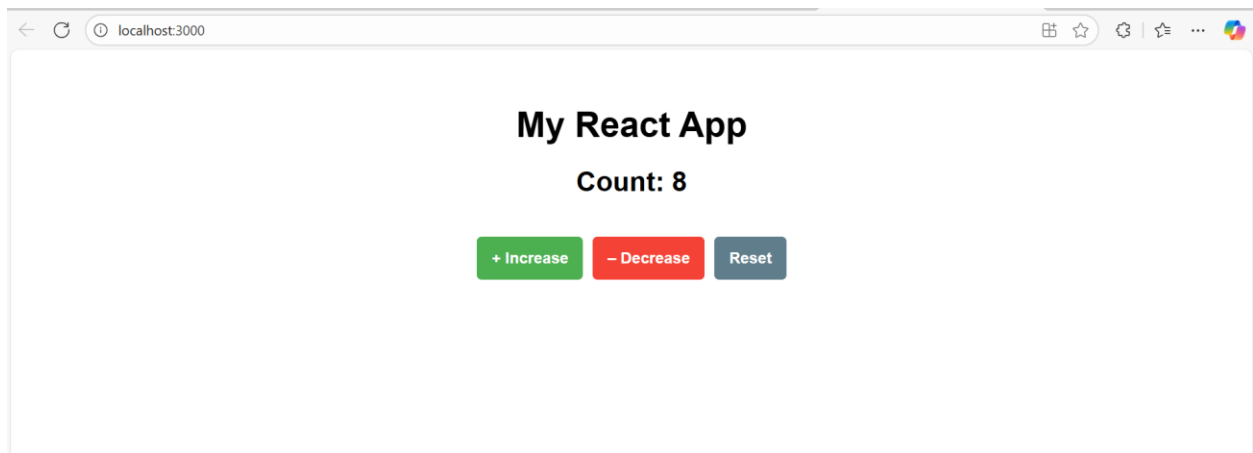
```
JS useCounter.js  JS Counter.js X  JS App.js
src > components > JS Counter.js > ...
3  function Counter() {
29    <div>
30      <button
31        onClick={increment}
32        style={{
33          padding: "10px 15px",
34          margin: "5px",
35          backgroundColor: "#4CAF50",
36          color: "white",
37          border: "none",
38          borderRadius: "5px",
39          fontWeight: "bold",
40          cursor: "pointer"
41        }}
42      >
43        + Increase
44      </button>
45      <button
46        onClick={decrement}
47        style={{
48          padding: "10px 15px",
49          margin: "5px",
50          backgroundColor: "#f44336",
51          color: "white",
52          border: "none",
53          borderRadius: "5px",
54          fontWeight: "bold",
55          cursor: "pointer"
56        }}
57      </button>
```

```
JS useCounter.js  JS Counter.js X  JS App.js
src > components > JS Counter.js > Counter
3  function Counter() {
55      cursor: "pointer"
56  }
57  >
58      Decrease
59  </button>
60  <button
61      onClick={reset}
62      style={{
63          padding: "10px 15px",
64          margin: "5px",
65          backgroundColor: "#607d8b",
66          color: "white",
67          border: "none",
68          borderRadius: "5px",
69          fontWeight: "bold",
70          cursor: "pointer"
71      }}
72  >
73      Reset
74  </button>
75  </div>
76  </div>
77  };
78  }
79
80  export default Counter;
81
```

App.js:

```
JS useCounter.js  JS Counter.js  JS App.js X
src > JS App.js > ...
1  // src/App.js
2  import React from "react";
3  import Counter from "../components/Counter";
4
5  function App() {
6      return (
7          <div className="App">
8              <Counter />
9          </div>
10     );
11 }
12
13 export default App;
14
```

We can Increase and decrease and reset the counter by clicking on the buttons:



Conclusion:

Today, I learned how to **create and use Custom Hooks in React**. These hooks allow us to write cleaner code by moving repetitive logic out of components. I practiced this by building a simple useCounter hook and using it in a component.