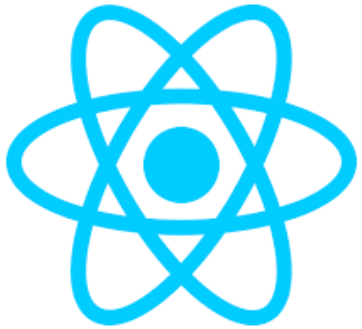


React JS & Hooks

MA003
ALYANI MAMAD B.

Agenda

What We Have Discussed Till Now?

Types Of Hooks

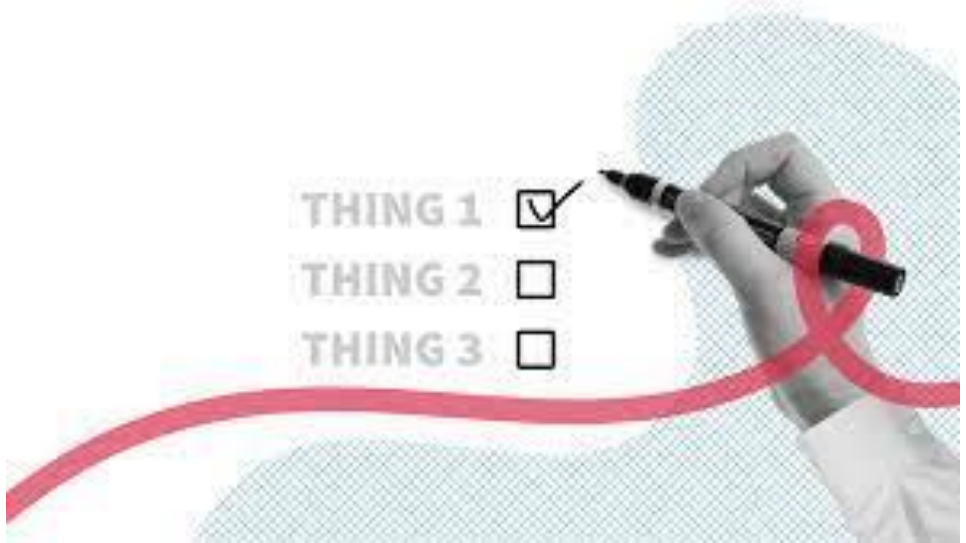
Introduction to useEffect

Effect Clean up

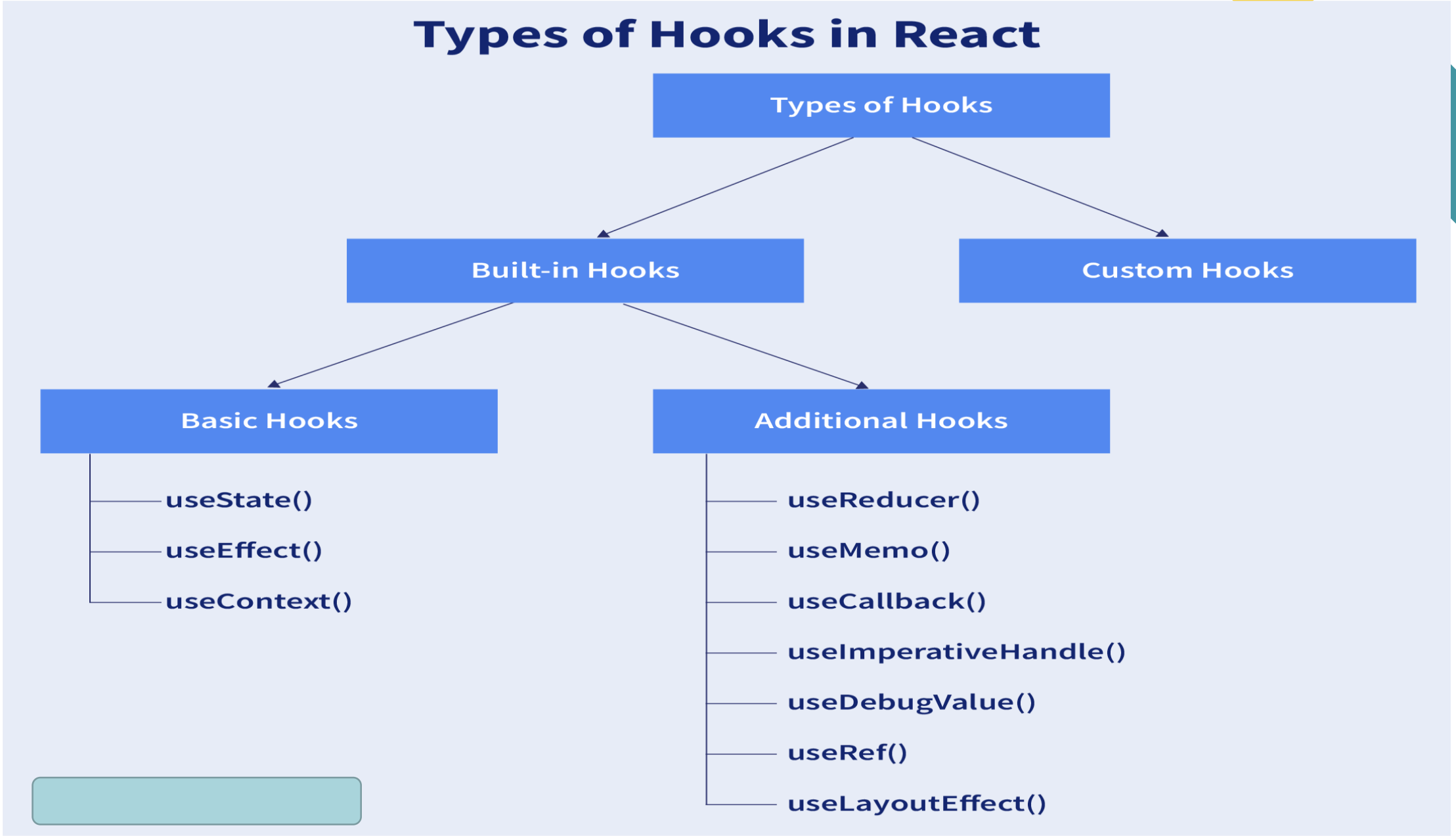
Conclusion

What We Have Discussed Till Now?

- Basic Introduction Of React Js
- Installation
- Some Prerequisite
- Introduction to Hooks and some Rules
- useState hook done



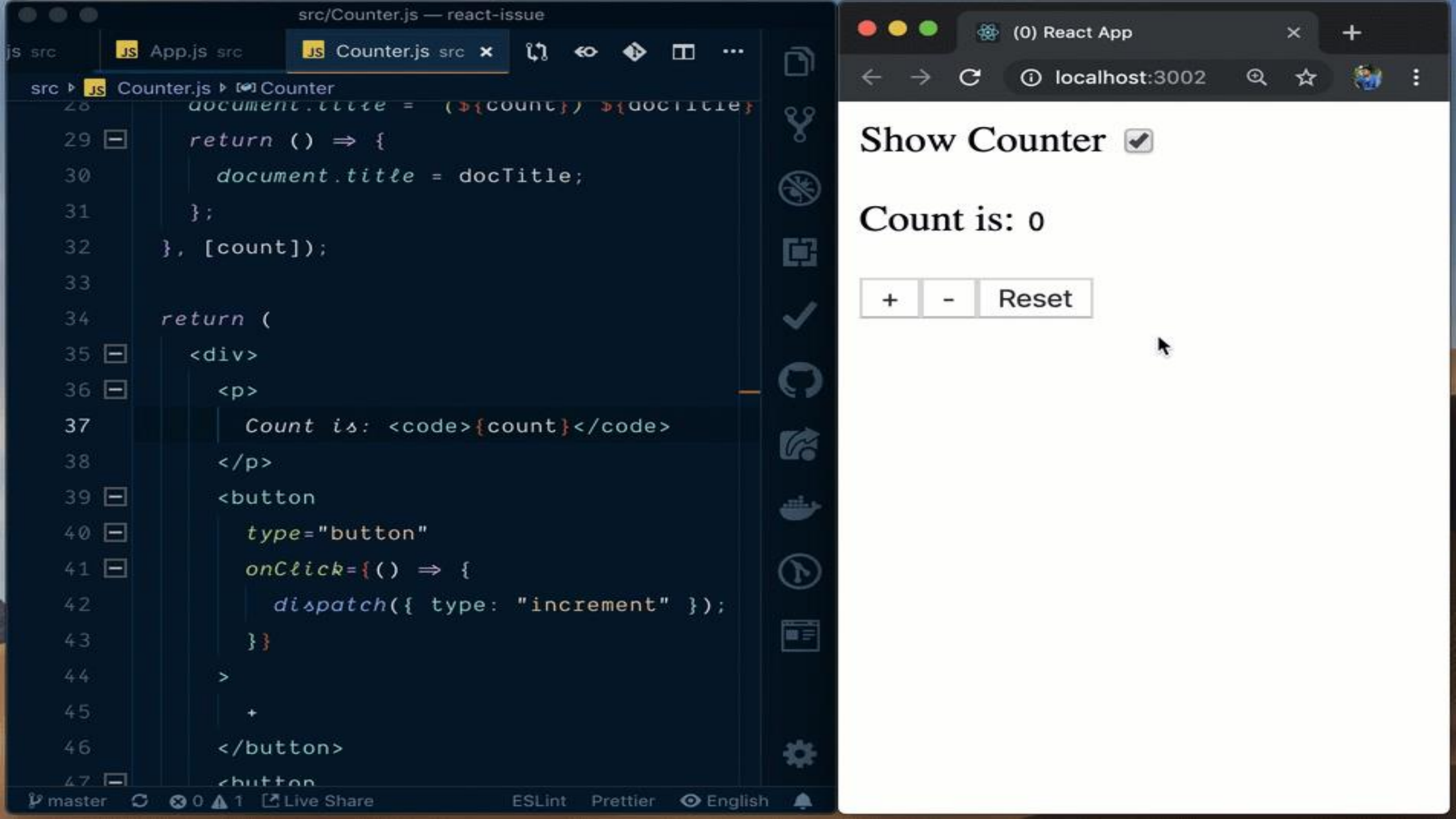
Types Of Hooks :



Introduction to useEffect

- The useEffect helps to perform side-effects(Outside current scope) in functional components.
- Automatically called when page LOADS.
- Eg : WHATSAPP chat count(chats(5)), Error-tracking.
- Used Outside the component to render data(API's).
- useEffect accepts two arguments. The second argument is optional.

- Syntax : `useEffect(<function>, <dependency>)`


src/Counter.js — react-issue
App.js src
Counter.js src
src ▸ Counter.js ▸ Counter
document.title = `(${count}) ${docTitle}`;
return () => {
document.title = docTitle;
};
}, [count]);
return (

Count is: <code>{count}</code>

<button
type="button"
onClick={() => {
dispatch({ type: "increment" });
}}
>
+
</button>
master
0 1
Live Share
ESLint
Prettier
English
(0) React App
localhost:3002
Show Counter
Count is: 0
+
-
Reset

Effect Cleanup

- Some effects require cleanup to reduce memory leaks.
- Timeouts, subscriptions, event listeners, and other effects that are no longer needed should be disposed
- We do this by including a return function at the end of the `useEffect` Hook.

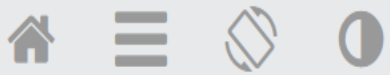
```
useEffect( () => {

  // perform a side effect

  return function () { /* clean up side effect */ };

}, [ ] );
```

clean-up function:
return a function to clean up after the effect. E.g. unsubscribe, stop timers, remove listeners, etc.



Result Size: 683 x 565

Get your own React server

```jsx
import { useState, useEffect } from "react";
import ReactDOM from "react-dom/client";

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

  useEffect(() => {
    let timer = setTimeout(() => {
    setCount((count) => count + 1);
  }, 1000);

  return () => clearTimeout(timer)
  }, []);

  return <h1>I've rendered {count} times!</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Timer />);

/*
Note: To clear the timer, we had to name it.
*/
```

localhost:3000

I have rendered 1 times!

ANY QUESTION?