

Twitter Thread by [Pratham](#)



[Pratham](#)
[@Prathkum](#)



React Hooks are the functions which "hook into" React state and lifecycle features from function components. Hooks allows you to manipulate state and other React feature without writing a class. Let's talk about widely used hook

useEffect hook at a glance ■■■■



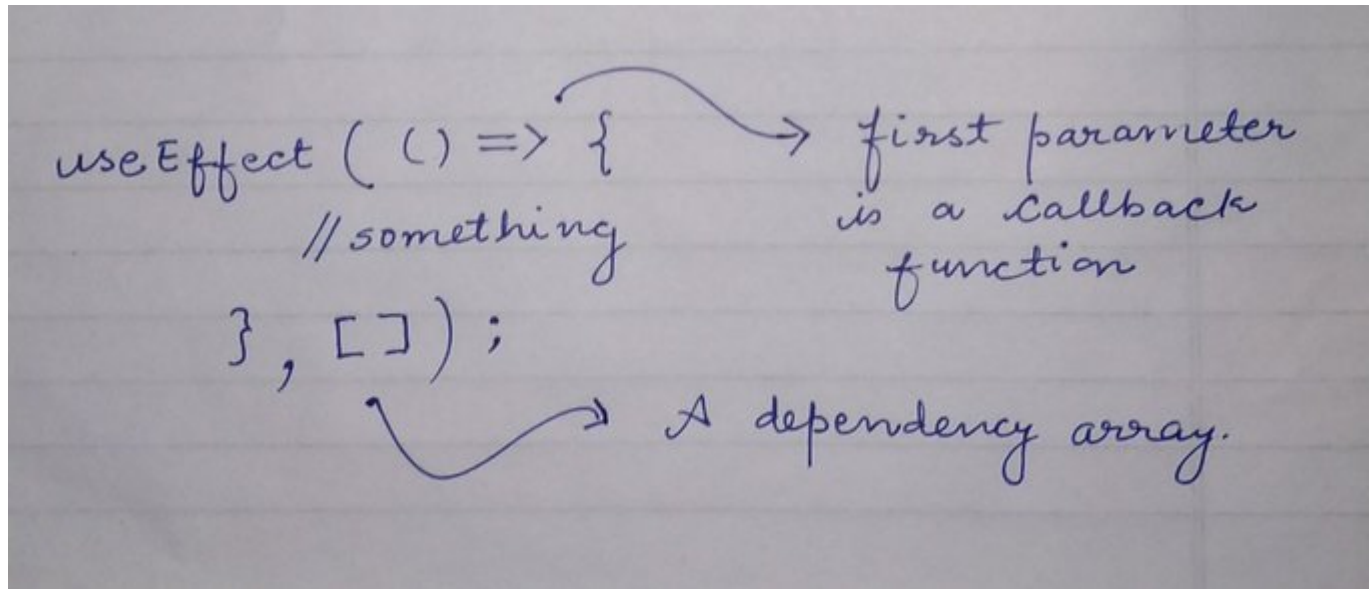
useEffect hook is the heart of React functional components

If you're familiar with class components then you might know that we have various lifecycle methods but in functional components, we don't have any lifecycle methods. Instead we have a powerful hook called useEffect

By using `useEffect`, you tell React that your component needs to do something after render. React will remember the function you passed (we'll refer to it as our "effect"), and call it later after performing the DOM updates

So let's start by understanding the syntax first

3/15



`useEffect` take two parameters, first is a function and second is an array.

The function inside the `useEffect` will run every single time component re-render. Consider this piece of code and check the output in next tweet

{ 4 / 15 }

```

function UseEffect() {

  const [render, setRender] = useState(0);

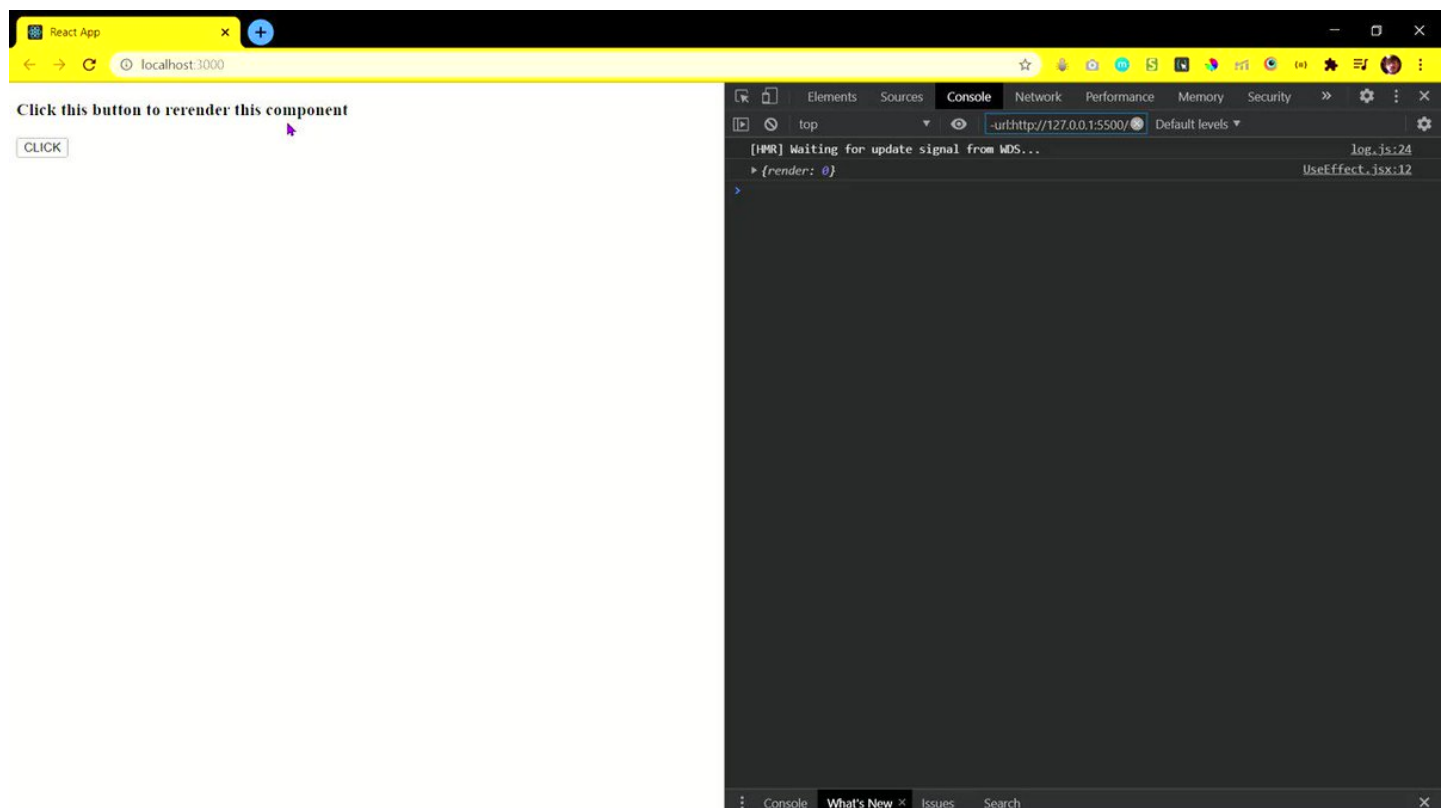
  // This will render everytime component re-rendered
  useEffect(() => console.log({ render }));

  return (
    <div>
      <h3>Click this button to rerender this component</h3>
      <button onClick={() => setRender((prevValue) => prevValue + 1)}>
        CLICK
      </button>
    </div>
  );
}

```

As you can see in the output the function is executed every single time my component re-render

{ 5 / 15 }



But let's say if I add some dependency in the array and pass the array as second parameter then useEffect will only run when the value of dependency array change.

For example, let me modify the code little bit so that you can understand it better

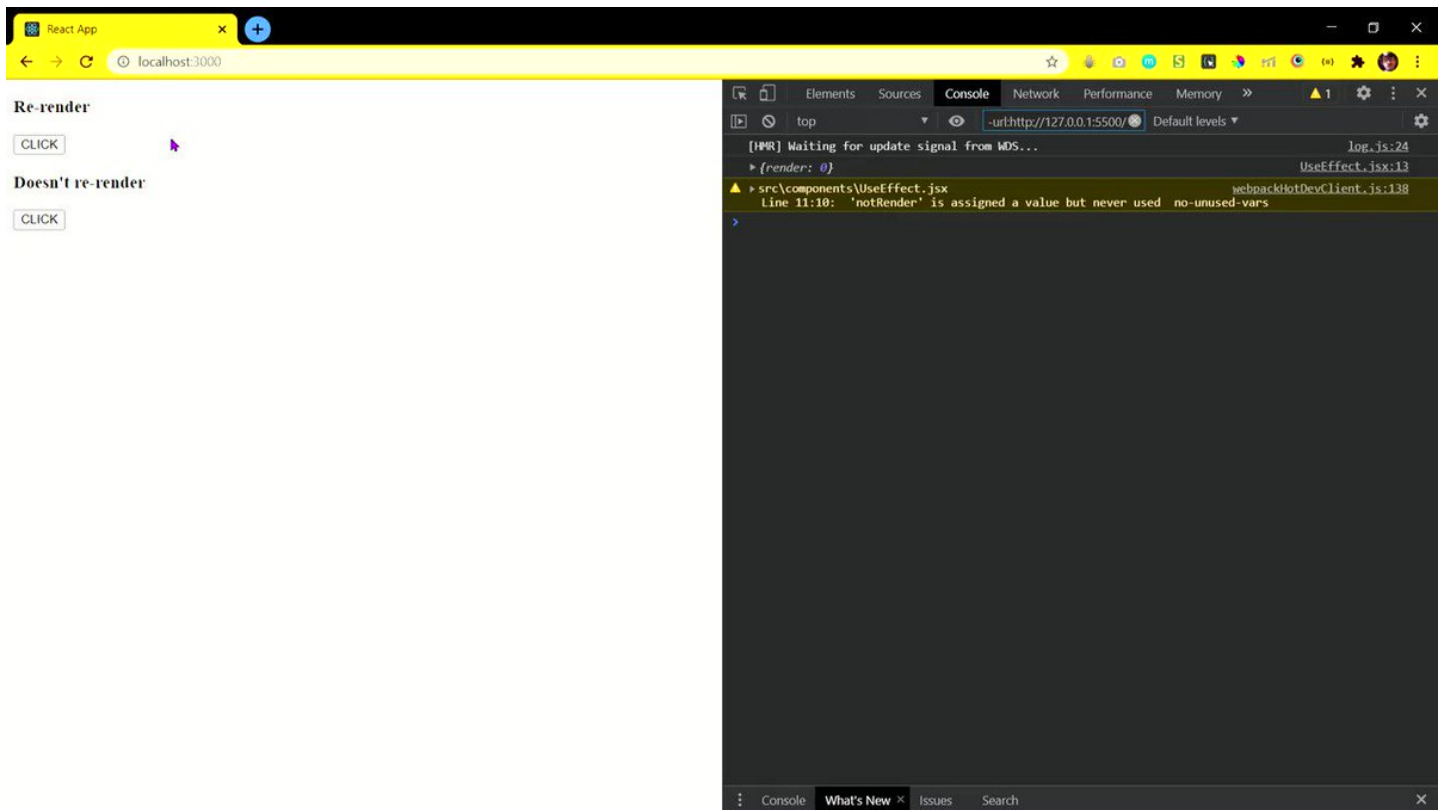
{ 6 / 15 }



```
function UseEffect() {  
  
  const [render, setRender] = useState(0);  
  const [notRender, setNotRender] = useState(0);  
  
  useEffect(() => console.log({ render }), [render]);  
  // Only runs when the value of render changes because I have passed render  
  in dependency array  
  
  return (  
    <div>  
      <h3>Re-render</h3>  
      <button onClick={() => setRender((prevValue) => prevValue + 1)}>  
        CLICK  
      </button>  
      <h3>Doesn't re-render</h3>  
      <button onClick={() => setNotRender((prevValue) => prevValue + 1)}>  
        CLICK  
      </button>  
    </div>  
  );  
}
```

As you can see when I click on the re-render button, our useEffect runs this is because I have passed render state inside dependency array

{ 7 / 15 }



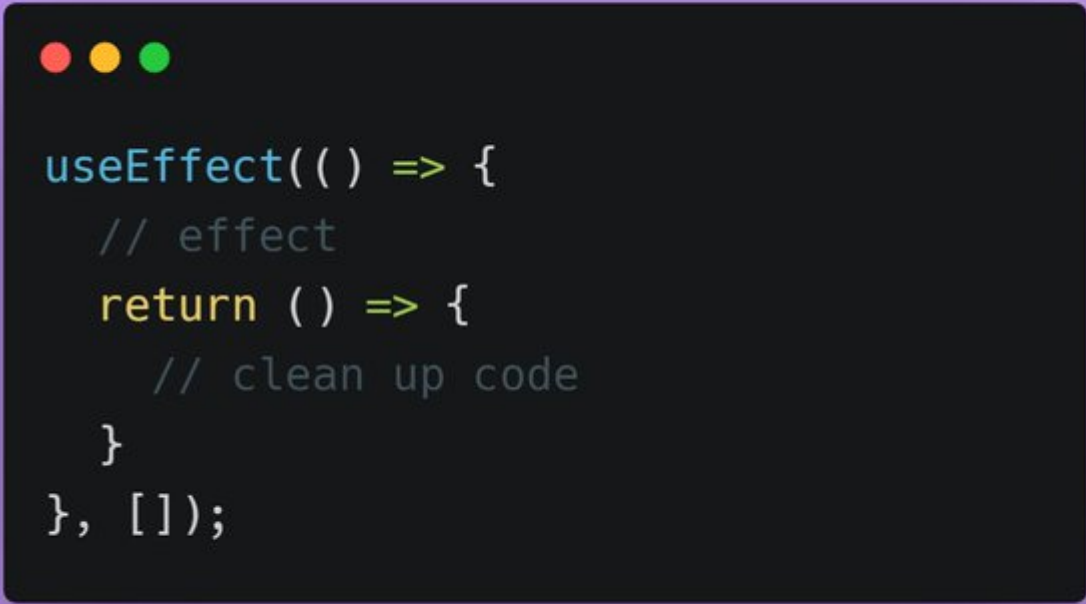
■ Here's an important thing to note is that if you pass an empty array then it will only run on once. No matter how many times you render your component, the useEffect will run only once because the value of empty array never going to change

{ 8 / 15 }

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

In useEffect we can also perform clean up

If we return a function within the method, this function perform basically a clean up of what we did last time.

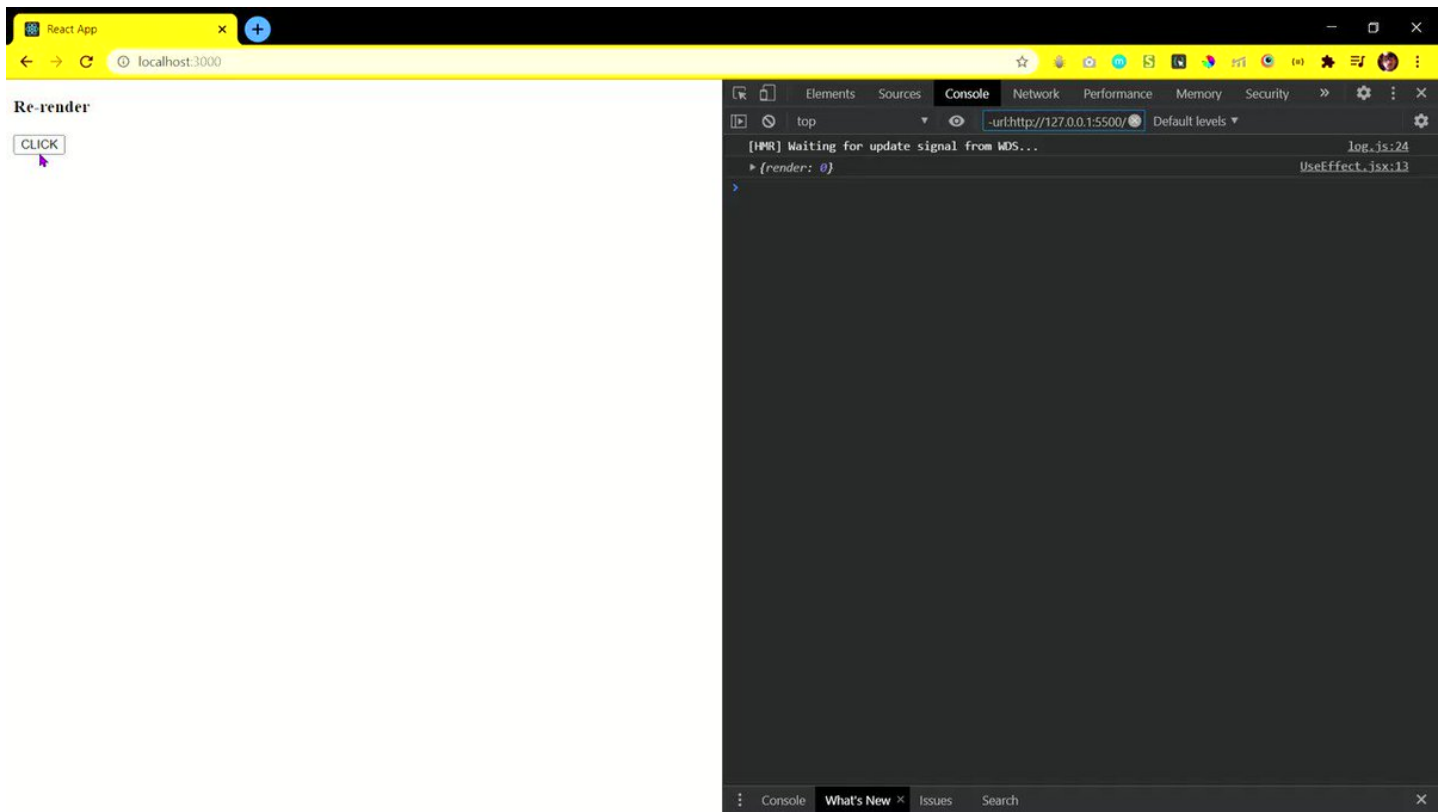


```
useEffect(() => {  
  // effect  
  return () => {  
    // clean up code  
  }  
}, []);
```

For example, consider this piece of code

```
useEffect(() => {  
  
  console.log({ render });  
  
  return () => {  
    console.log("I'm cleanup function");  
  };  
}, [render]);
```

Everytime I click the button, first our useEffect perform clean up then run the effect function



So far we have discussed the basics of useEffect.

Let's build something useful using it. We will be using useEffect for fetching some COVID data

{ 11 / 15 }



We will print total number of confirmed COVID cases of a specific country enter by user in the input field.

On the basis of the value entered by user we will store that in "country" and change that value in our API link

{ 12 / 15 }

- Make an input field
- on form submit, store the input value in "country"

Print the confirmed cases on screen as simple as that

check the entire code

{ 13 / 15 }



Check out the source code on this link in more accessible form

<https://t.co/r9G6i7EAIW>

{ 14 / 15 }

Awesome! I think we have covered everything related to useEffect hook. It is little tough so try to play around with the code. I hope you like this thread ❤️■

Peace out ■