



# REACT STATE

State, hooks, useState



# But what is 'State'?

It's neither a difficult concept, nor a React-specific one as it turns out!

# let's understand what the word "State" actually means - **outside of programming.**

- If I'm drinking coffee, my current state is that I'm drinking coffee.
- We could also narrow that down: I might be moving my coffee mug to my mouth, so my state is that I'm holding the coffee mug and that I'm lifting it up to my mouth. My state also is that my mouth is opened.
- So "State" is not just one thing.
- Think of water and ice. What changes their 'state'?

# And in programming, it's **basically the same!**

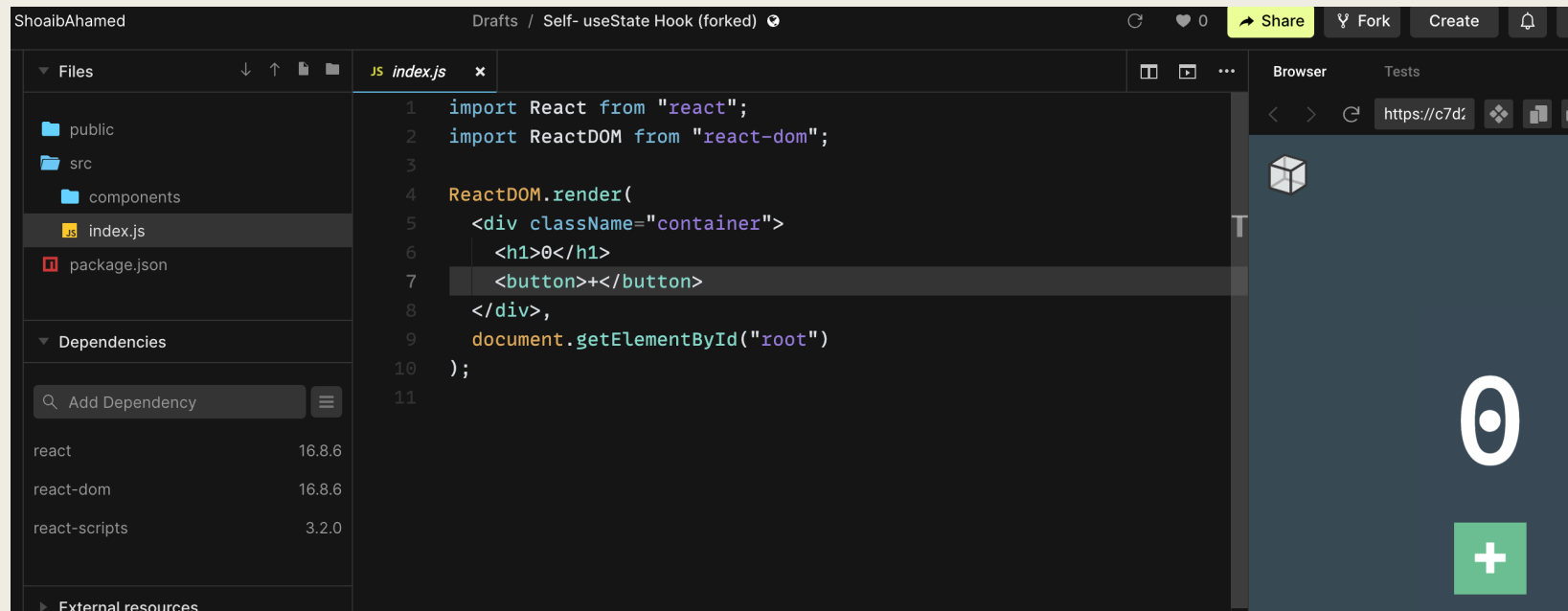
- In a web application, the overall state might be, that a modal overlay is displayed, asking the visitor for input (e.g. an authentication overlay).
- That's the state of the web application!
- Of course, we could also split that into smaller parts: The state of the modal, is that it's opened. The state of the form in the modal is that it's empty. And so on.

# But what is "State" in Programming Specifically?

- The state (i.e. "current snapshot") of your program or of a part of it - depending on how you look at it.
- It's the combination of all those individual states (like "modal is open" and "user is not authenticated", for example) that makes up the overall program state.

# useState Hook

Go to this codesandbox - <https://codesandbox.io/s/students-usestate-hook-forked-z4rsrz?file=/src/index.js> and check the index.js file .



The screenshot shows a Codesandbox workspace titled "ShoaibAhamed" with a draft named "Self- useState Hook (forked)". The left sidebar displays the file structure with folders "public" and "src", and files "index.js" and "package.json". The "Dependencies" section lists "react" (16.8.6), "react-dom" (16.8.6), and "react-scripts" (3.2.0). The main editor shows the content of "index.js":

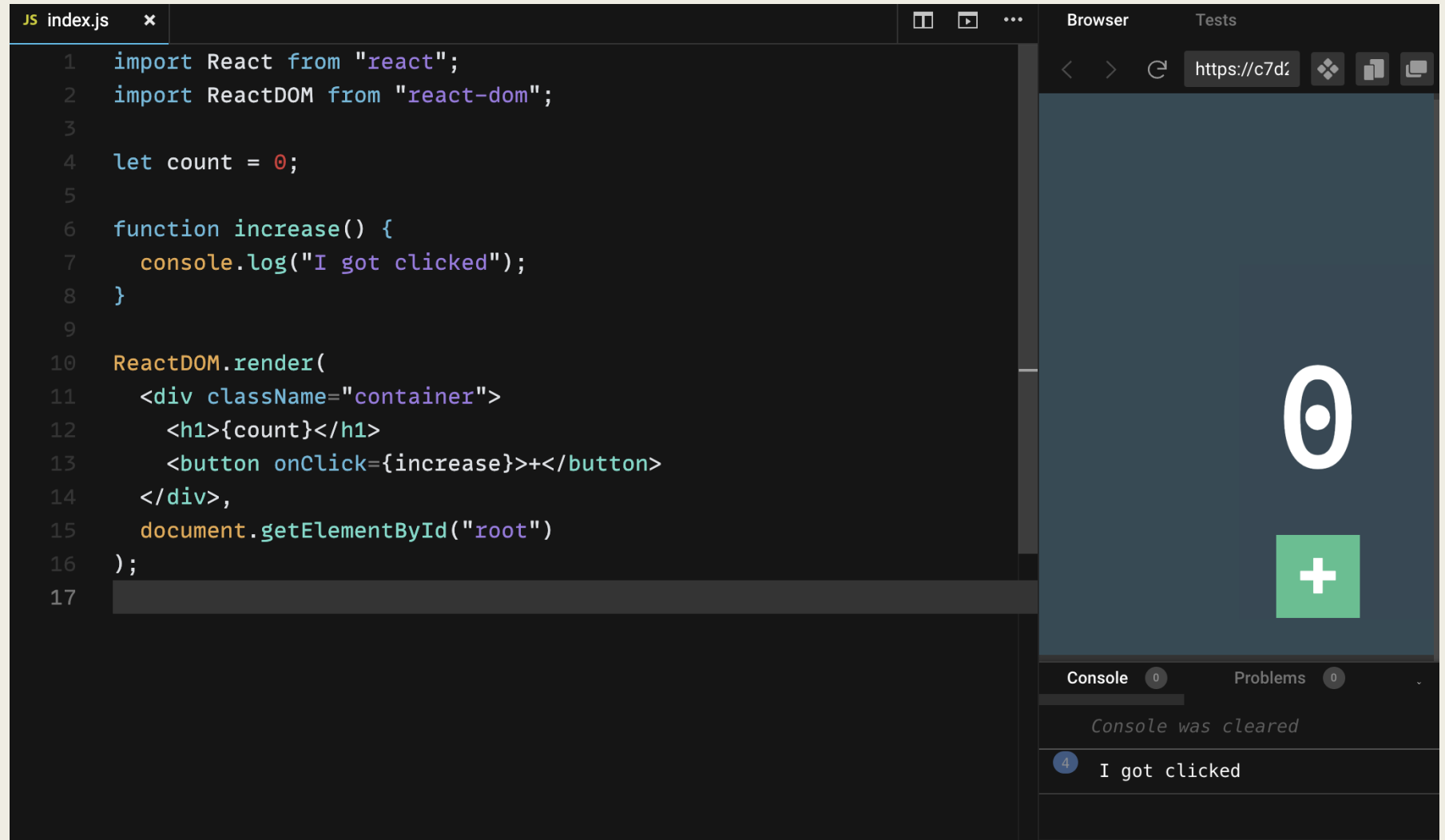
```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 ReactDOM.render(
5   <div className="container">
6     <h1>0</h1>
7     <button>+</button>
8   </div>,
9   document.getElementById("root")
10 );
11
```

On the right, a browser preview shows a dark blue background with a large white "0" and a green square button with a white "+" sign at the bottom right.

# How to activate the button to increase the value by one once the button is clicked?

- First declare a new variable `count` and replace `h1` content with `{}`.
- Add `onClick` attribute to the button
- Create a function named `increase` and assign that to `onClick = {increase}`

This -



The image shows a development environment with a code editor on the left and a browser on the right. The code editor displays a JavaScript file named `index.js` with the following content:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 let count = 0;
5
6 function increase() {
7   console.log("I got clicked");
8 }
9
10 ReactDOM.render(
11   <div className="container">
12     <h1>{count}</h1>
13     <button onClick={increase}>+</button>
14   </div>,
15   document.getElementById("root")
16 );
17
```

The browser on the right shows the rendered application at the URL `https://c7d4...`. The application displays a large white digit `0` on a dark blue background. Below the digit is a green square button with a white plus sign `+`. The browser's console at the bottom shows a message `I got clicked` with a blue circle containing the number `4` next to it. The console also shows the message `Console was cleared`.



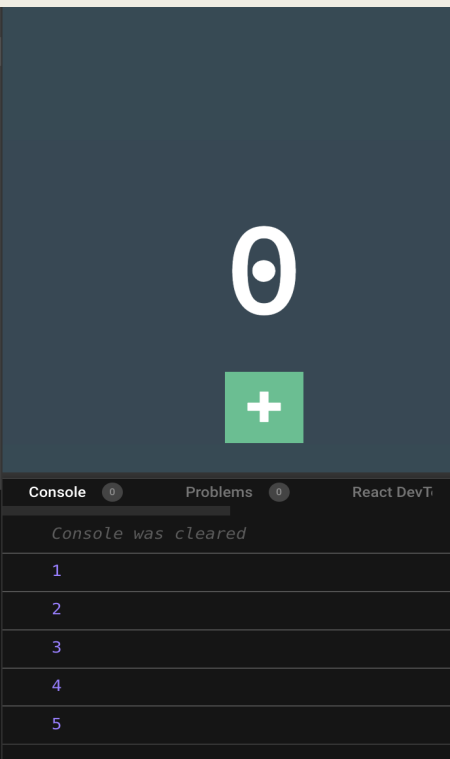
# But how can we increase the value of our count variable and in the browser?

count++ and logging the count

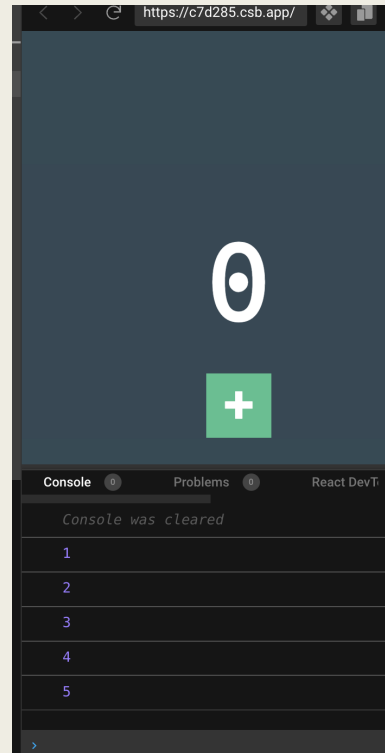
```
let count = 0;

function increase() {
  count++;
  console.log(count);
}

ReactDOM.render(
  <div className="container">
    <h1>{count}</h1>
    <button onClick={increase}>+</button>
  </div>,
  document.getElementById("root")
);
```



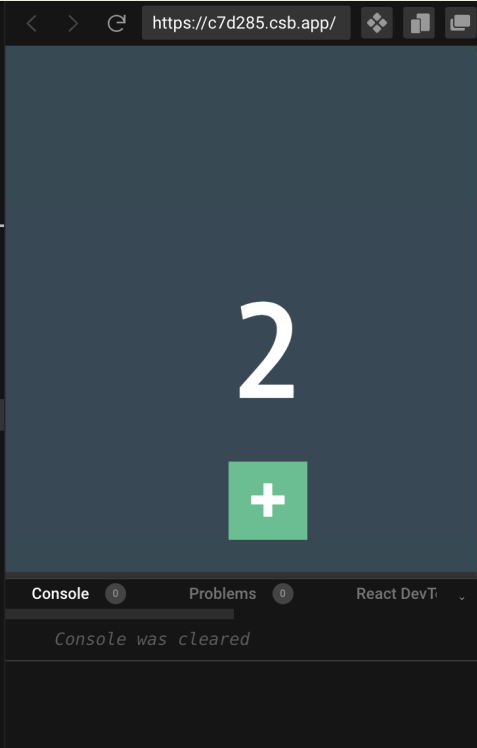
# But still nothing changing the user Interface!



# How to update user interface then?

One (inefficient) way is to replace the `console.log` with the `ReactDOM.render` code.

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 let count = 0;
5
6 function increase() {
7   count++;
8   ReactDOM.render(
9     <div className="container">
10       <h1>{count}</h1>
11       <button onClick={increase}>+</button>
12     </div>,
13     document.getElementById("root")
14   );
15 }
16
17 ReactDOM.render(
18   <div className="container">
19     <h1>{count}</h1>
20     <button onClick={increase}>+</button>
21   </div>,
22   document.getElementById("root")
23 );
```



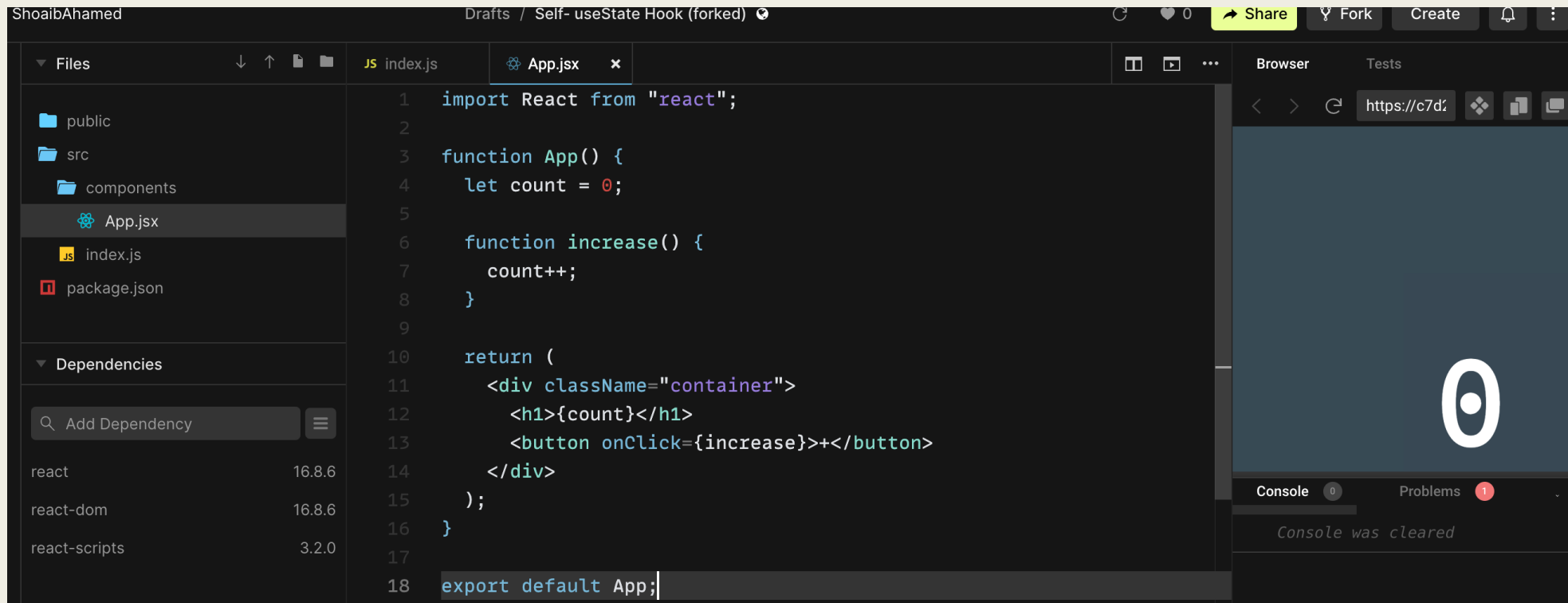
# Every time the button is clicked it's re-rendering everything on the screen!

- There are repetition of code!

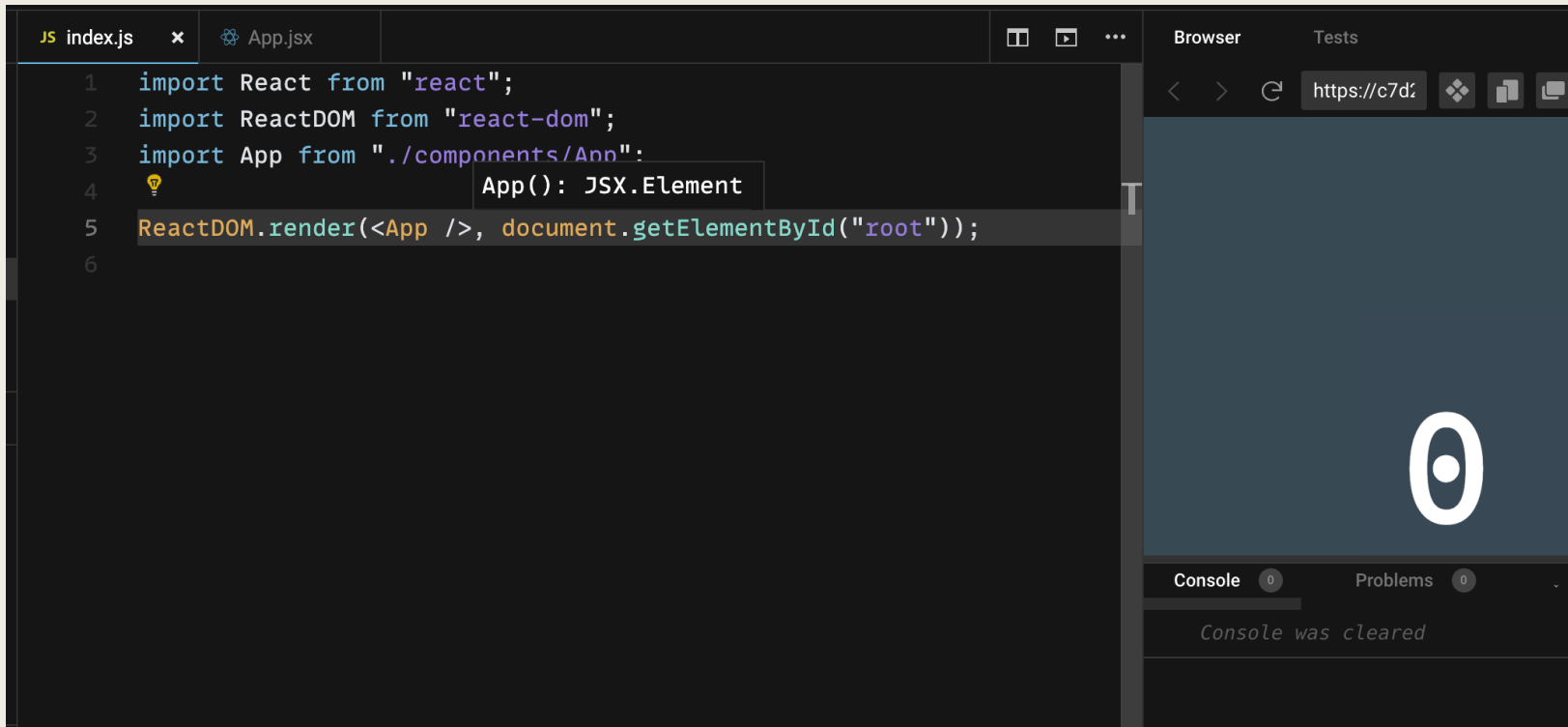
# How to solve this in React? useState Hooks!

- Hooks are functions that let you “hook into” React state and lifecycle features from **function components**. More [here](#).
- In order to use hooks, we have to have a functional components.
- Let's create an App component in order to use hook.
- Then move all of the codes from index.js to App.jsx

# App.jsx



# Index.js



The image shows a code editor with two tabs: 'index.js' and 'App.jsx'. The 'index.js' tab is active, displaying the following code:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import App from "../components/App";
4 
5 ReactDOM.render(<App />, document.getElementById("root"));
6
```

A tooltip is visible over the `<App />` tag, showing `App(): JSX.Element`. To the right of the code editor is a browser window with the address bar showing `https://c7d...`. The browser displays a dark blue background with a large white number '0' in the center. Below the browser window, there is a 'Console' tab with a message that says 'Console was cleared'.

# Setting useState() : Either way

In App.jsx file

```
import React from "react";

function App() {
  const state = React.useState();
}
```


```
JS index.js  App.jsx x
1  import React, { useState } from "react";
2
3  function App() {
4    const state = useState();
5
```



# console.logging state will -

Return an Array with two items 1. undefined 2. a function

```
3 function App() {  
4   const state = useState();  
5  
6   console.log(state);  
7  
8   function increase() {  
9     // count++;  
10  }  
11  
12  return (  
13    <div className="container">  
14      <h1>0</h1>  
15      <button onClick={increase}>+</button>  
16    </div>  
17  );  
18 }  
19  
20 export default App;
```

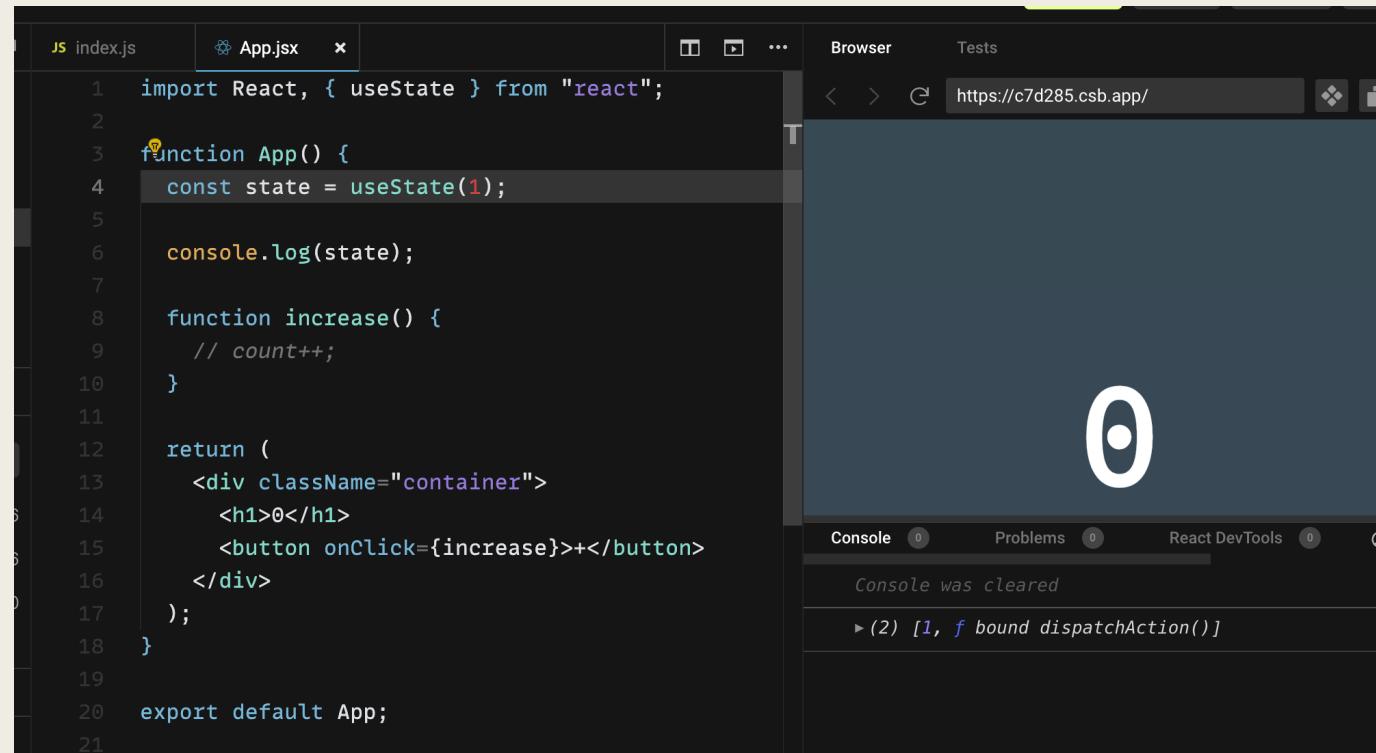


Console 0 Problems 0 React DevTools 0

Console was cleared

▼ (2) [undefined, f bound dispatchAction()]  
 0: undefined  
 ► 1: f bound dispatchAction() {}

# Initial state : passing 1 instead undefined

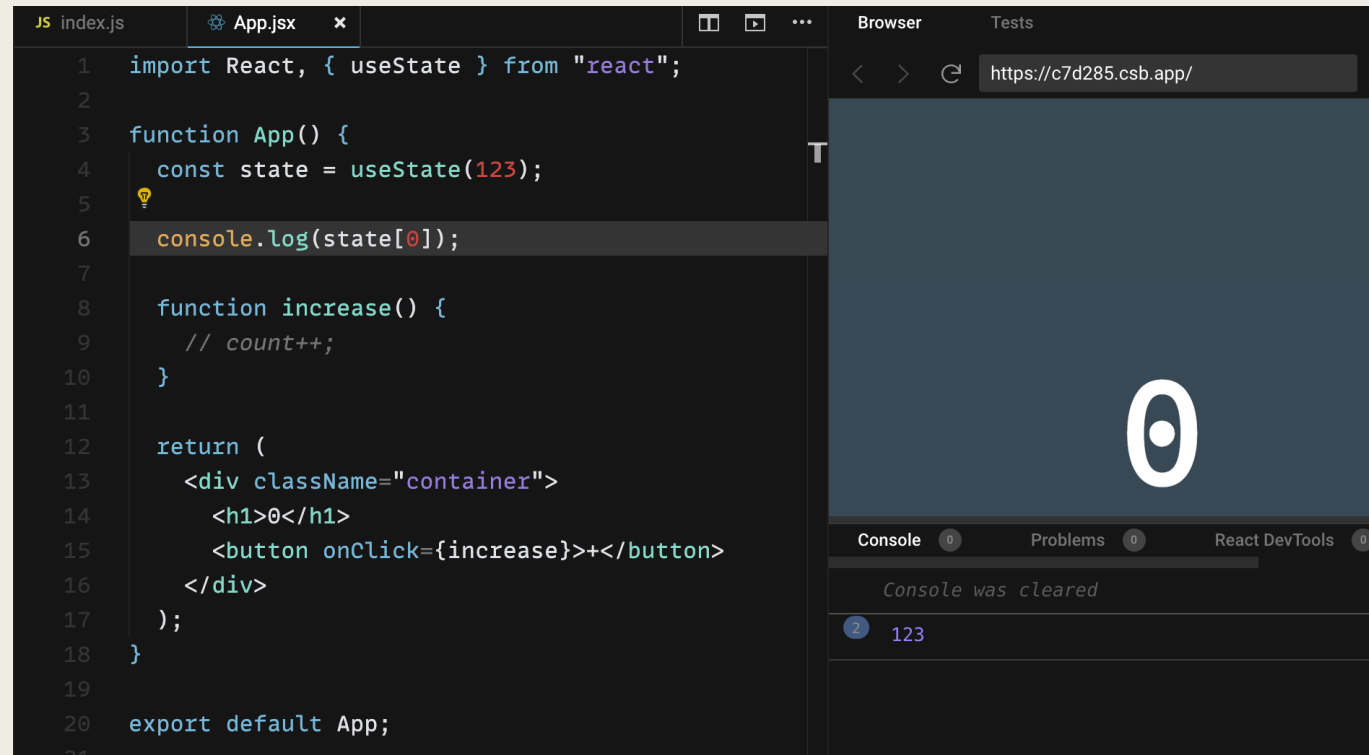


```
1 import React, { useState } from "react";
2
3 function App() {
4   const state = useState(1);
5
6   console.log(state);
7
8   function increase() {
9     // count++;
10  }
11
12  return (
13    <div className="container">
14      <h1>0</h1>
15      <button onClick={increase}>+</button>
16    </div>
17  );
18 }
19
20 export default App;
```

The browser window shows the URL <https://c7d285.csb.app/> and displays a large white '0' on a dark blue background.

The console shows the message: `[1, f bound dispatchAction()]`.

# State[0] holds the initial value



The screenshot shows a code editor with a file named `App.jsx` and a browser window displaying the application. The code in `App.jsx` is as follows:

```
1 import React, { useState } from "react";
2
3 function App() {
4   const state = useState(123);
5
6   console.log(state[0]);
7
8   function increase() {
9     // count++;
10  }
11
12  return (
13    <div className="container">
14      <h1>0</h1>
15      <button onClick={increase}>+</button>
16    </div>
17  );
18 }
19
20 export default App;
```

The browser window shows the URL `https://c7d285.csb.app/` and displays a large white number `0` on a dark blue background. The React DevTools console is visible at the bottom, showing the message `Console was cleared` and a log entry `123` with a blue circle icon next to it.

Replacing h1 content with {state[0]} will update in the browser.

```
function increase() {  
  // count++;  
}  
  
return (  
  <div className="container">  
    <h1>{state[0]}</h1>  
    <button onClick={increase}>+</button>  
  </div>  
);
```

123

Console

0

Problems

0

React DevTools

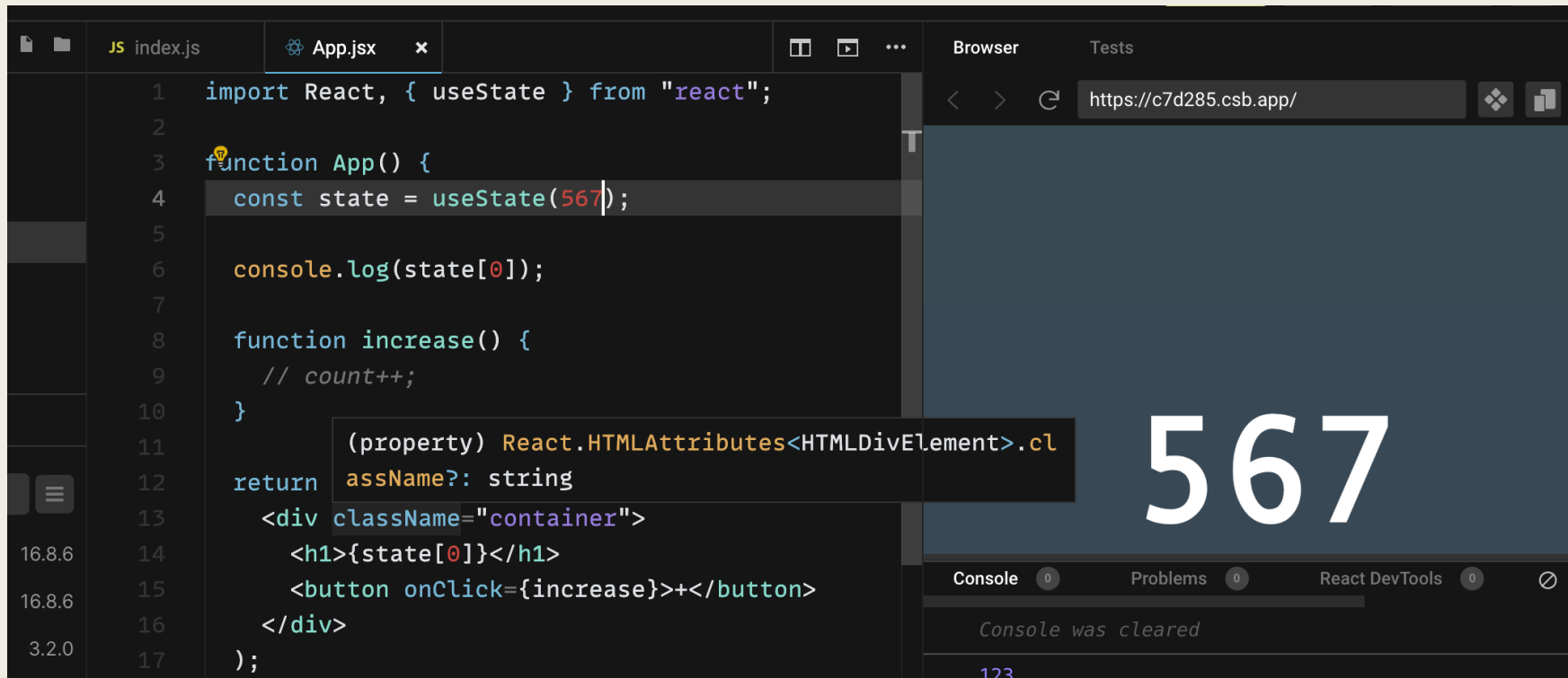
0

⌵

Console was cleared

123

# Changing value in useState() immediately updates value in browser! (So cool!)



The image shows a development environment with VS Code on the left and a web browser on the right. The VS Code editor has two tabs: 'index.js' and 'App.jsx'. The 'App.jsx' tab is active, showing the following code:

```
1 import React, { useState } from "react";
2
3 function App() {
4   const state = useState(567);
5
6   console.log(state[0]);
7
8   function increase() {
9     // count++;
10  }
11
12  return (
13    <div className="container">
14      <h1>{state[0]}</h1>
15      <button onClick={increase}>+</button>
16    </div>
17  );
```

A tooltip is visible over the code, showing the type signature: `(property) React.HTMLAttributes<HTMLDivElement>.className?: string`. The web browser on the right shows the URL `https://c7d285.csb.app/` and displays the number `567` in a large white font on a dark blue background. The bottom of the browser window shows the 'Console' tab with the message 'Console was cleared' and the number '123'.

# JS Destructuring – Destructure a complex structure – Objects and Arrays

```
const color = [9, 132, 227];  
console.log(color[0]);  
const [red, blue, green] = [9, 132, 227];  
console.log(blue);
```

```
return (  
  <div className="container">  
    <h1>{state[0]}</h1>  
    <button onClick={increase}>+</button>
```

567

Console 0

Problems 2

React DevTools 0



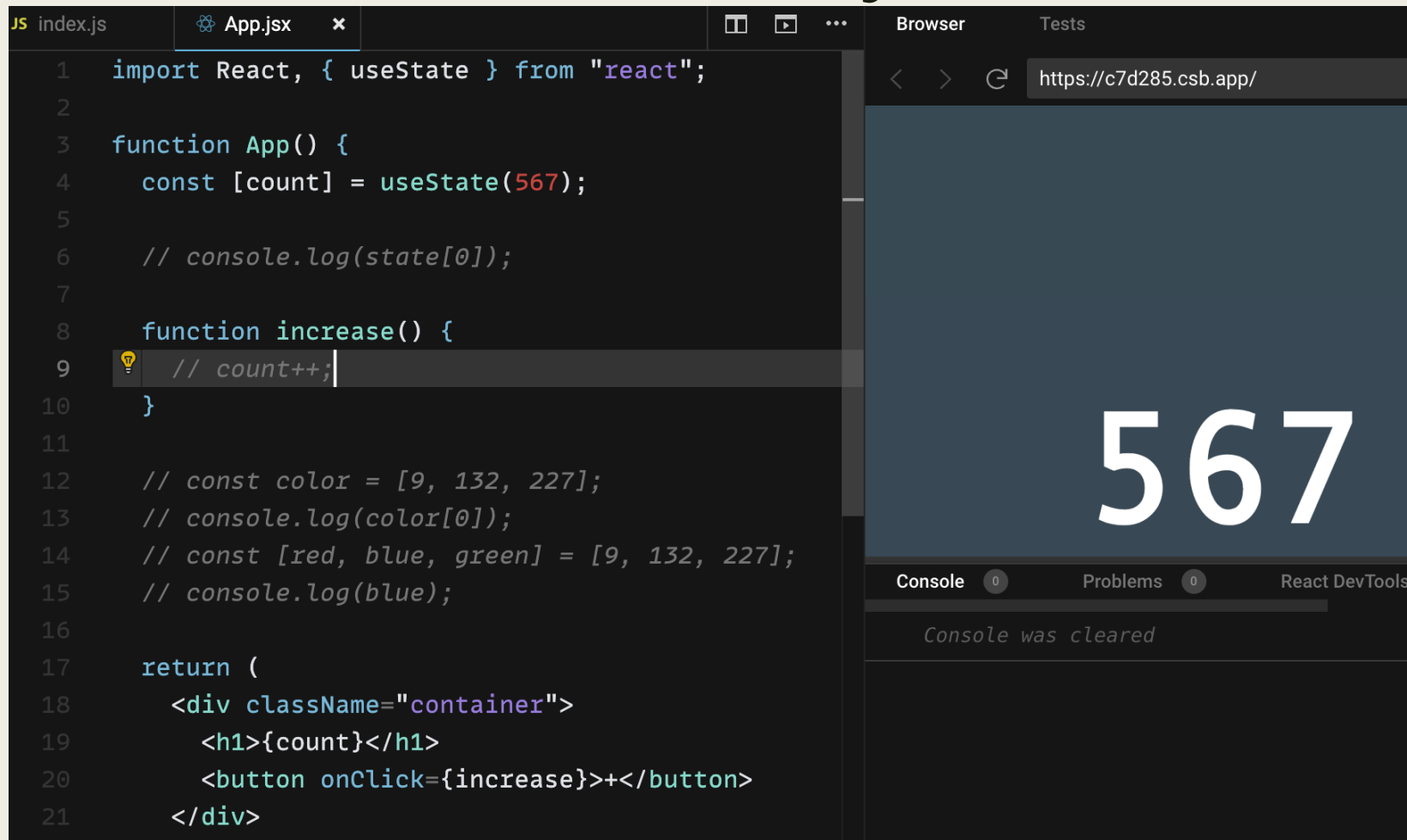
Console was cleared

567

9

132

# Destructuring state : [count] as we know it returns an array-



The image shows a development environment with a code editor on the left and a browser on the right. The code editor displays the following JavaScript code in `App.jsx`:

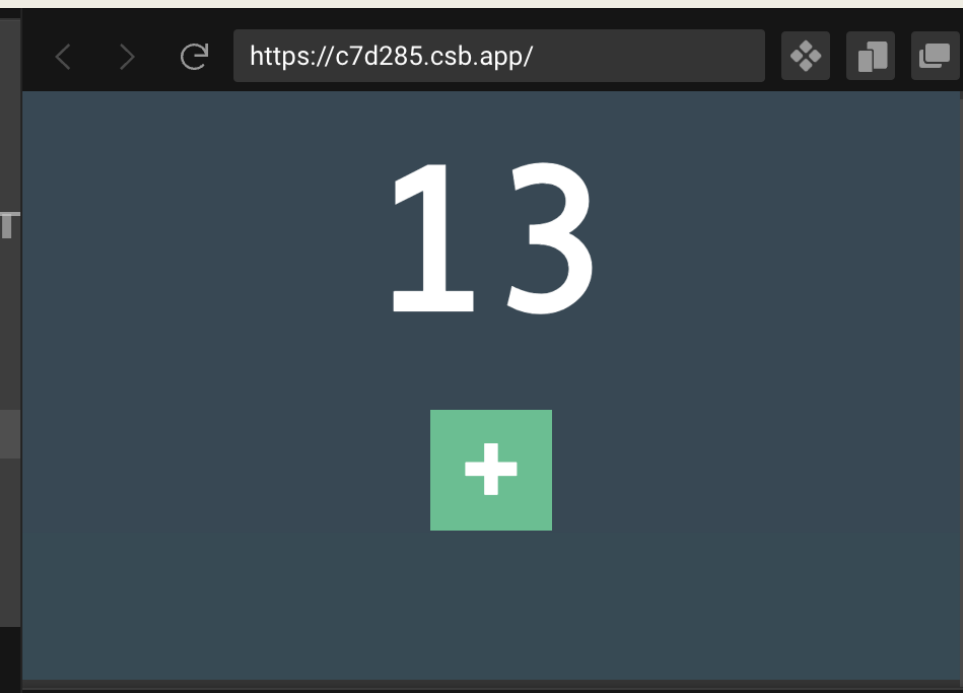
```
1 import React, { useState } from "react";
2
3 function App() {
4   const [count] = useState(567);
5
6   // console.log(state[0]);
7
8   function increase() {
9     // count++;
10  }
11
12  // const color = [9, 132, 227];
13  // console.log(color[0]);
14  // const [red, blue, green] = [9, 132, 227];
15  // console.log(blue);
16
17  return (
18    <div className="container">
19      <h1>{count}</h1>
20      <button onClick={increase}>+</button>
21    </div>
```

The browser window shows the URL `https://c7d285.csb.app/` and displays the number `567` in a large white font on a dark blue background. The bottom of the browser window shows the React DevTools interface with tabs for Console, Problems, and React DevTools. The Console tab is active and shows the message "Console was cleared".

How do we change the initial value( count)?  
- using setCount function ( the second item of array).

First trying setting setCount(30)

```
1 import React, { useState } from "react";
2
3 function App() {
4   const [count, setCount] = useState(0);
5
6   // console.log(state[0]);
7
8   function increase() {
9     setCount(count + 1);
10  }
11
12  // const color = [9, 132, 227];
13  // console.log(color[0]);
14  // const [red, blue, green] = [9, 132, 227];
```



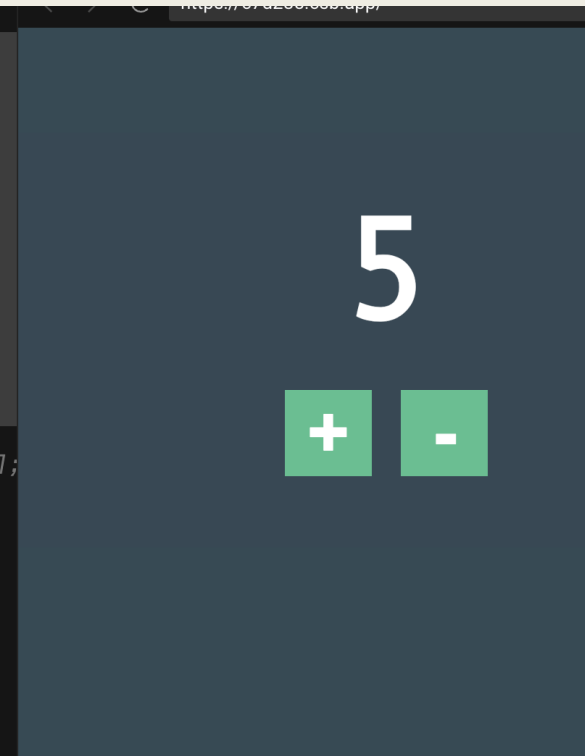


- Codes up to this point can be found here - <https://codesandbox.io/s/self-usestate-hook-forked-c7d285?file=/src/components/App.jsx>

Challenge : create a button that  
decreases the values.

# Solution:

```
5
6 // console.log(state[0]);
7
8 function increase() {
9   setCount(count + 1);
10 }
11
12 function decrease() {
13   setCount(count - 1);
14 }
15
16 // const color = [9, 132, 227];
17 // console.log(color[0]);
18 // const [red, blue, green] = [9, 132, 227];
19 // console.log(blue);
20
21 return (
22   <div className="container">
23     <h1>{count}</h1>
24     <button onClick={increase}>+</button>
25     <button onClick={decrease}>-</button>
26   </div>
```



Learn useEffect hooks -

[https://www.w3schools.com/REACT/react\\_useeffect.asp](https://www.w3schools.com/REACT/react_useeffect.asp)

More on different react hooks - <https://react.dev/reference/react>

# Assignment

*//Go here - <https://codesandbox.io/s/students-assignment-usestate-hook-practice-forked-lf9n3n?file=/src/index.js:158-658>*

*//1. Given that you can get the current time using:*

```
let time = new Date().toLocaleTimeString();
```

```
console.log(time);
```

*//Show the latest time in the <h1> when the Get Time button is pressed.*

*//2. Given that you can get code to be called every second using the setInterval method.*

*Can you get the time in your <h1> to update every second?*

*//e.g. uncomment the code below to see Hey printed every second.*

```
// function sayHi() {
```

```
// console.log("Hey");
```

```
// }
```

```
// setInterval(sayHi, 1000);
```

# Materials credit : Our gratitude to -

- App Brewery
- W3schools
- Udemy
- Programming Hero
- Youtube
- Github
- React Programming: The Big Nerd Ranch Guide