



EVENT HANDLING



We will learn -

- How to detect when users interact with the app and
- How to render different things based on those events.
- Go here - <https://codesandbox.io/s/event-handling-in-react-forked-msr7np>

App.jsx – h1, input, button

```
App.jsx x
1 import React from "react";
2
3 function App() {
4   return (
5     <div className="container">
6       <h1>Hello</h1>
7       <input type="text" placeholder="What's your name?" />
8       <button>Submit</button>
9     </div>
10  );
11 }
12
13 export default App;
14
```

Browser Tests

< > ↺ https://6v7385.csb.app/

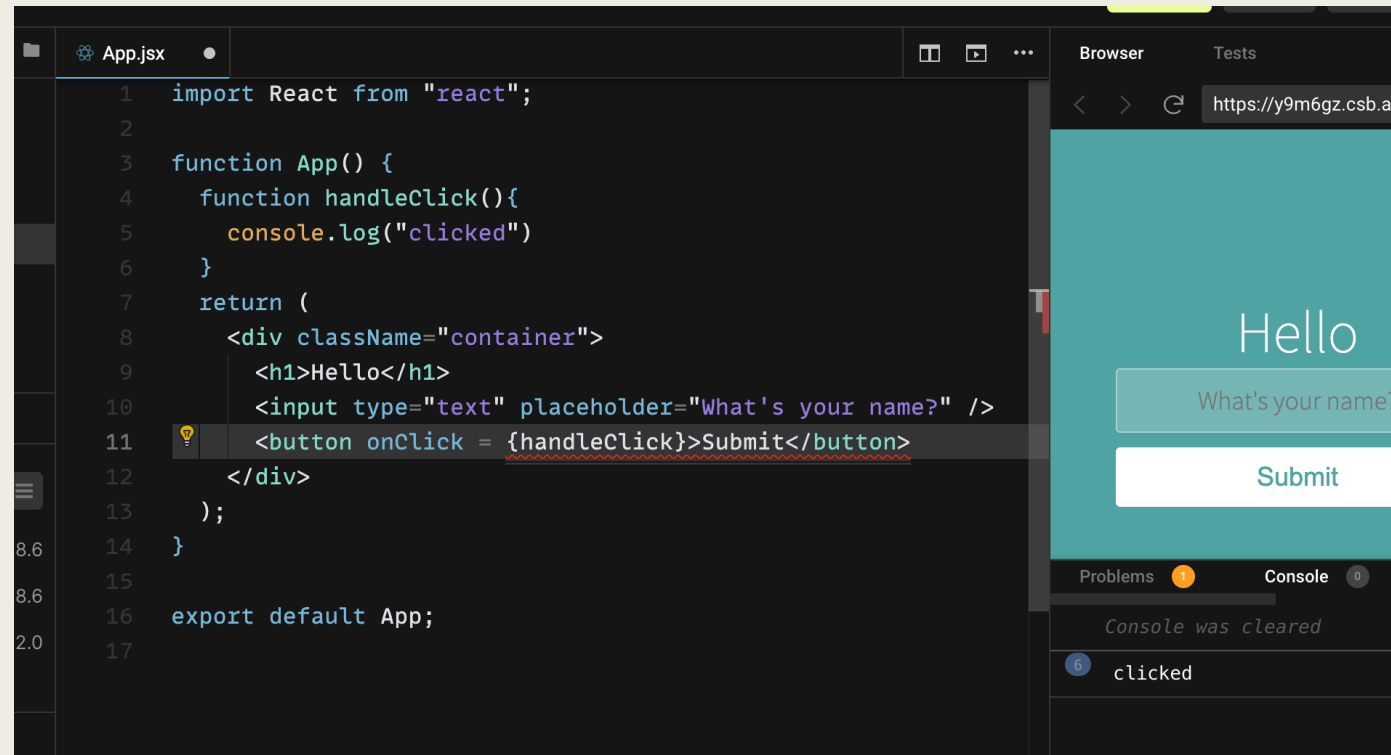
⌵

Hello

What's your name?

Submit

Activating button : onClick, handleClick()



The screenshot displays a code editor with a file named `App.jsx`. The code defines a function `App()` that returns a JSX element. Inside the JSX, there is a `button` with an `onClick` prop set to `handleClick`. The `handleClick` function is defined to log "clicked" to the console. The browser preview on the right shows the rendered UI: a teal background with the text "Hello", a text input field with the placeholder "What's your name?", and a "Submit" button. The console at the bottom shows the message "clicked" after the button was clicked.

```
1 import React from "react";
2
3 function App() {
4   function handleClick(){
5     console.log("clicked")
6   }
7   return (
8     <div className="container">
9       <h1>Hello</h1>
10      <input type="text" placeholder="What's your name?" />
11      <button onClick = {handleClick}>Submit</button>
12    </div>
13  );
14 }
15
16 export default App;
17
```

Browser: <https://y9m6gz.csb.a>

Hello

What's your name?

Submit

Problems 1 Console 0

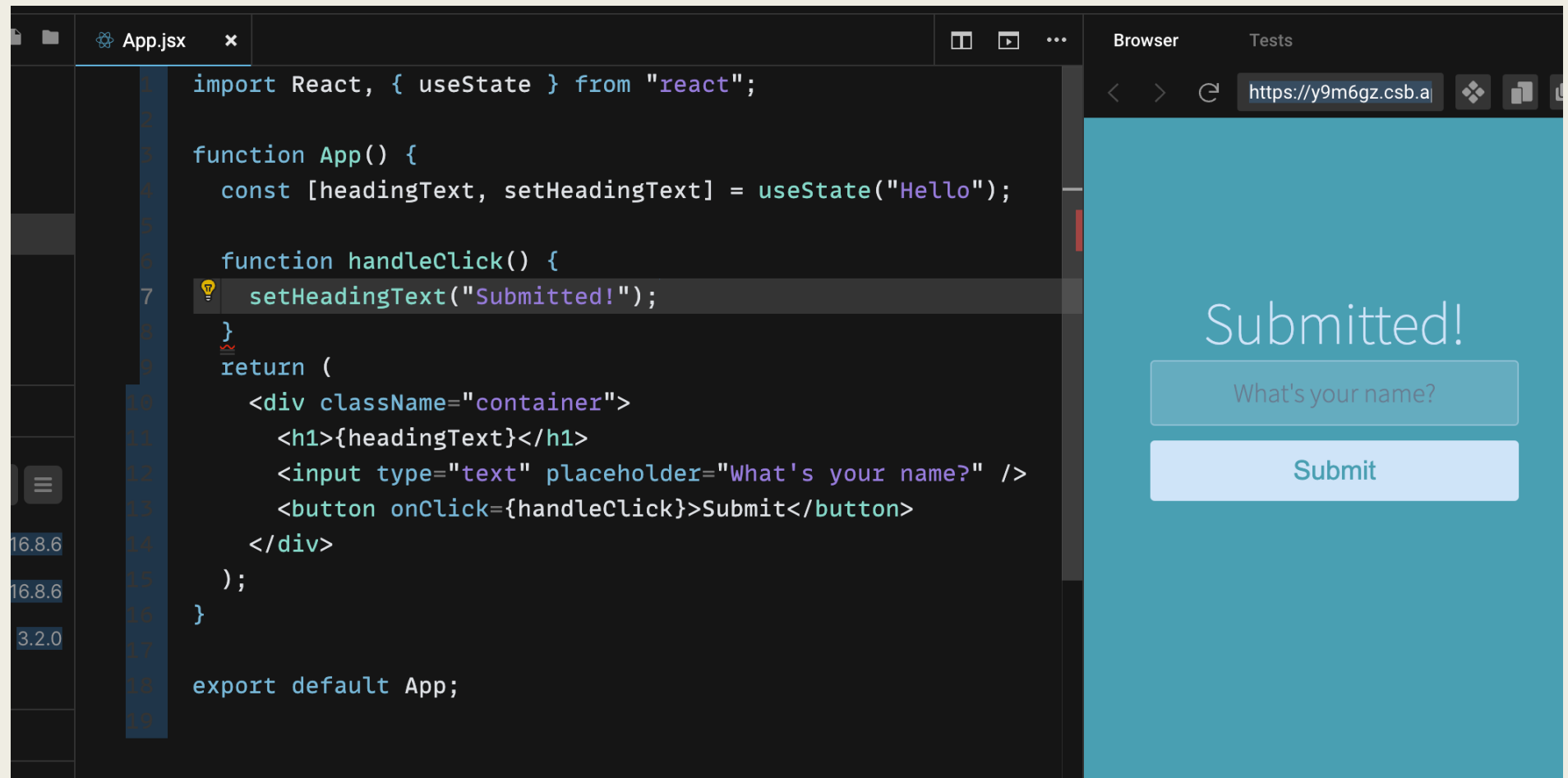
Console was cleared

6 clicked

How can we change the content of h1
when the button is clicked?

const [headingText, setHeadingText] = useState();

- Use useState() hooks, declare the starting state and a function that change that starting state.



The image shows a code editor window with a file named 'App.jsx' and a browser window displaying the rendered application. The code in the editor uses the useState hook to manage a heading state. The browser shows the result of the state change.

```
1 import React, { useState } from "react";
2
3 function App() {
4   const [headingText, setHeadingText] = useState("Hello");
5
6   function handleClick() {
7     setHeadingText("Submitted!");
8   }
9   return (
10    <div className="container">
11      <h1>{headingText}</h1>
12      <input type="text" placeholder="What's your name?" />
13      <button onClick={handleClick}>Submit</button>
14    </div>
15  );
16 }
17
18 export default App;
```

The browser window shows the URL <https://y9m6gz.csb.a> and displays the text "Submitted!" in a large font. Below it is a text input field with the placeholder "What's your name?" and a "Submit" button.

What are the different events?

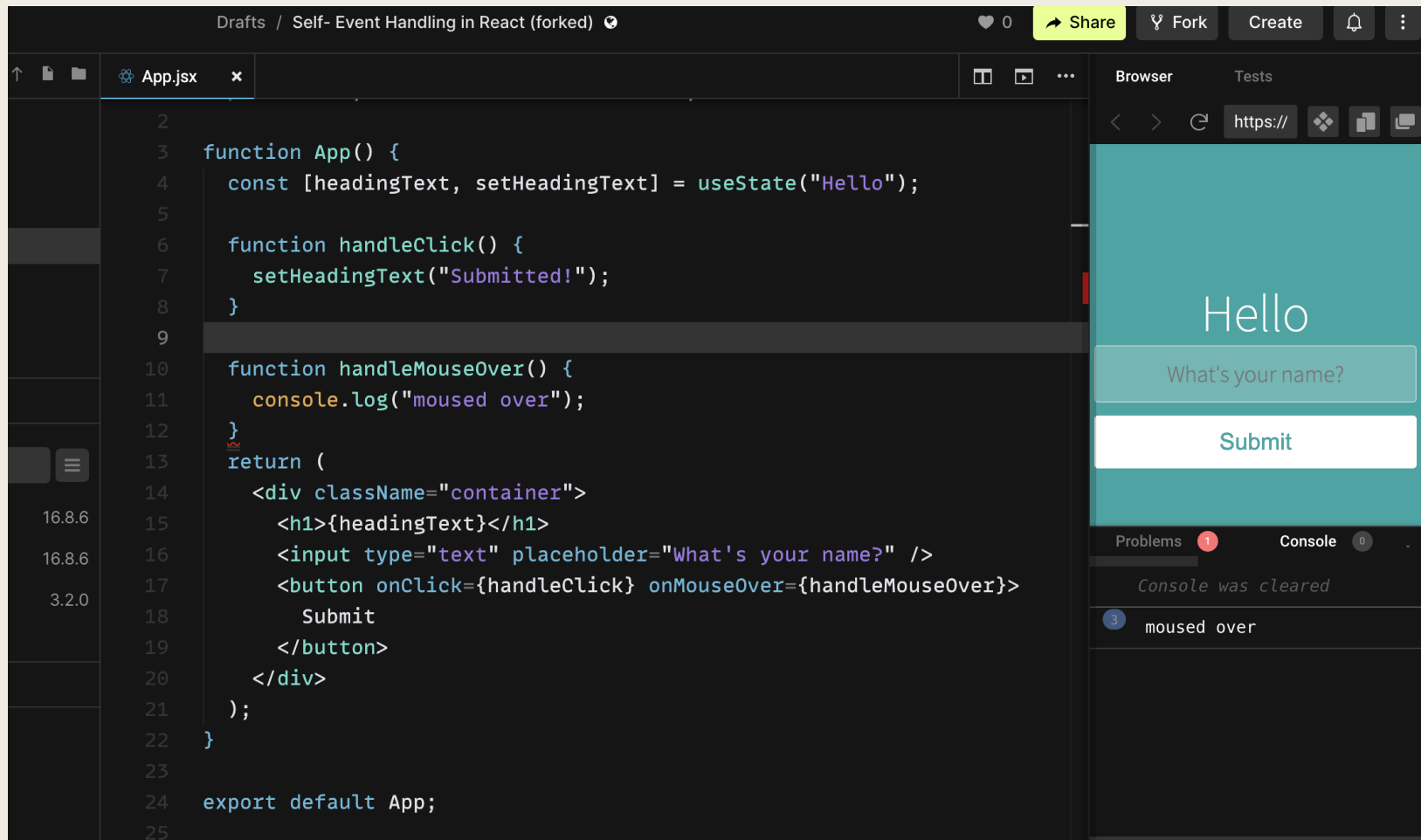
- List - https://www.w3schools.com/tags/ref_eventattributes.asp

Challenge :

- Use onmouseout and onmouseover events to change the background color of submit button when hovered over.

Solution

- Use onMouseOver = handleMouseOver



The screenshot shows a code editor with a file named 'App.jsx'. The code defines a function 'App()' that uses 'useState' to manage 'headingText'. It includes two event handlers: 'handleClick' which sets 'headingText' to 'Submitted!', and 'handleMouseOver' which logs 'moused over' to the console. The JSX returns a container with an 'h1' tag for 'headingText', a text input with placeholder 'What's your name?', and a 'Submit' button. The button's 'onClick' is set to 'handleClick' and its 'onMouseOver' is set to 'handleMouseOver'. The right sidebar shows a browser preview of the application with the heading 'Hello' and the input field. The console at the bottom shows the log message 'moused over'.

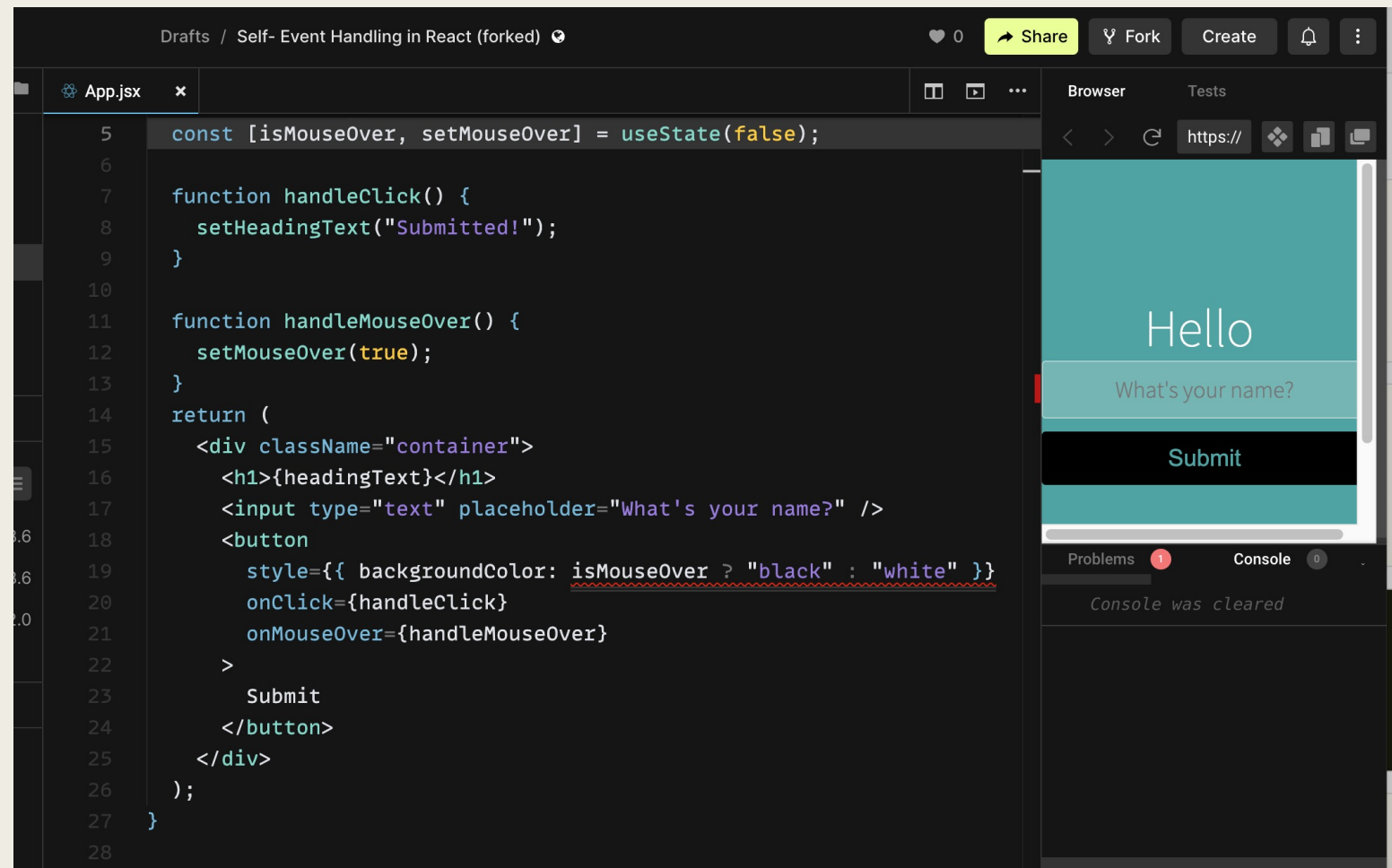
```
2
3 function App() {
4   const [headingText, setHeadingText] = useState("Hello");
5
6   function handleClick() {
7     setHeadingText("Submitted!");
8   }
9
10  function handleMouseOver() {
11    console.log("moused over");
12  }
13  return (
14    <div className="container">
15      <h1>{headingText}</h1>
16      <input type="text" placeholder="What's your name?" />
17      <button onClick={handleClick} onMouseOver={handleMouseOver}>
18        Submit
19      </button>
20    </div>
21  );
22 }
23
24 export default App;
25
```

Use State to change the background color

- `const [isMouseOver, setMouseOver] = useState(false);`
- Starting value is false assuming when the page loads, mouse will not be on the button.
- Once the event is triggered `handleMouseOver()` sets `SetMouseOver(true)`;
- Also set a `backgroundColor : "white"`

```
<input type="text" placeholder="What's your name" />
<button
  style={{ backgroundColor: "white" }}
  onClick={handleClick}
  onMouseOver={handleMouseOver}
>
  Submit
</button>
```

Background color changed via mouseover

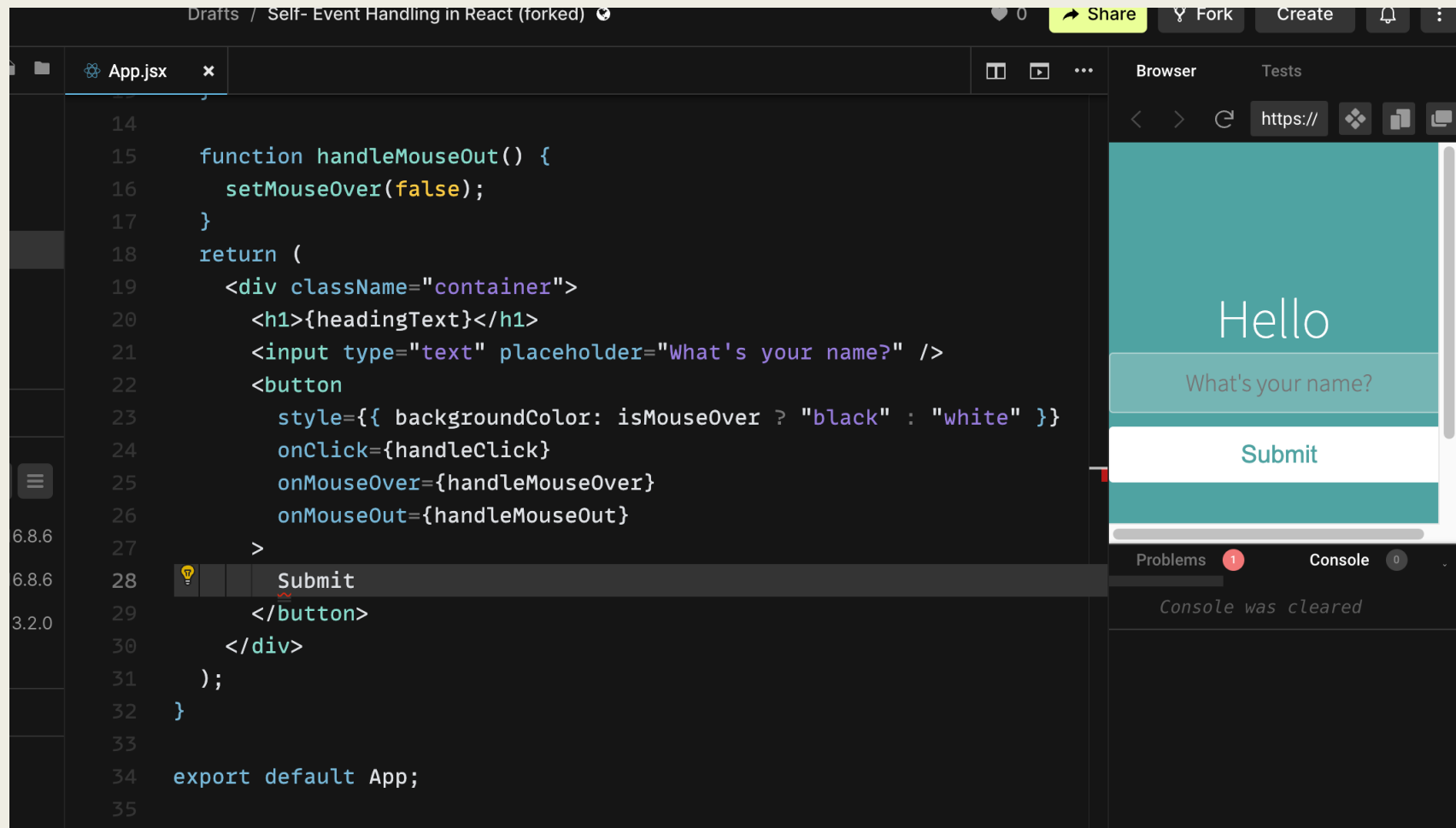


The screenshot shows a code editor with a file named 'App.jsx'. The code defines a state variable `isMouseOver` using `useState(false)`. It includes two functions: `handleClick` which sets `headingText` to 'Submitted!', and `handleMouseOver` which sets `isMouseOver` to `true`. The JSX returns a `div` with `className='container'` containing an `h1` with `headingText`, a text input with placeholder 'What's your name?', and a `button`. The button's `style` is set to `{background-color: isMouseOver ? 'black' : 'white'}`, and it has `onClick={handleClick}` and `onMouseOver={handleMouseOver}` props. The button text is 'Submit'. To the right, a browser preview shows the rendered UI: a teal header with 'Hello', a teal input field with 'What's your name?', and a black button with 'Submit' in white text. Below the browser, the 'Problems' and 'Console' panels are visible, with the console showing 'Console was cleared'.

```
5  const [isMouseOver, setMouseOver] = useState(false);
6
7  function handleClick() {
8    setHeadingText("Submitted!");
9  }
10
11 function handleMouseOver() {
12   setMouseOver(true);
13 }
14 return (
15   <div className="container">
16     <h1>{headingText}</h1>
17     <input type="text" placeholder="What's your name?" />
18     <button
19       style={{ backgroundColor: isMouseOver ? "black" : "white" }}
20       onClick={handleClick}
21       onMouseOver={handleMouseOver}
22     >
23       Submit
24     </button>
25   </div>
26 );
27 }
```

How to change back to white color
when mouse is out?

Solution : onMouseOut is set to false



The screenshot shows a code editor with a file named 'App.jsx'. The code defines a function `handleMouseOut()` that sets `isMouseOver` to `false`. It then returns a JSX element with a container div containing a heading, a text input, and a submit button. The button's `style.backgroundColor` is set to `isMouseOver ? 'black' : 'white'`. The `onClick`, `onMouseOver`, and `onMouseOut` props are all set to their respective handler functions. A right sidebar shows a browser preview of the application, displaying 'Hello', the text input, and the submit button. The button is currently white. Below the browser preview, a 'Problems' panel shows 1 error and a 'Console' panel shows 'Console was cleared'.

```
14
15 function handleMouseOut() {
16   setMouseOver(false);
17 }
18 return (
19   <div className="container">
20     <h1>{headingText}</h1>
21     <input type="text" placeholder="What's your name?" />
22     <button
23       style={{ backgroundColor: isMouseOver ? "black" : "white" }}
24       onClick={handleClick}
25       onMouseOver={handleMouseOver}
26       onMouseOut={handleMouseOut}
27     >
28       Submit
29     </button>
30   </div>
31 );
32 }
33
34 export default App;
35
```

- Learn more about conditional (ternary operator) - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_operator

- The full code can be viewed as - <https://codesandbox.io/s/self-event-handling-in-react-forked-y9m6gz?file=/src/components/App.jsx>