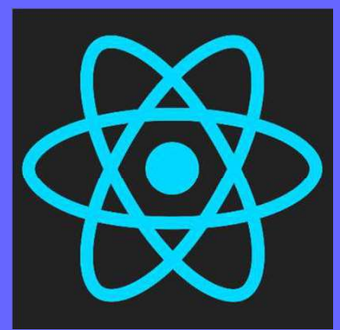
State in React

Objectives

After completing this lesson, you should be able to do the following:

- Basics in State
- setState
- State, Events and Managed Controls

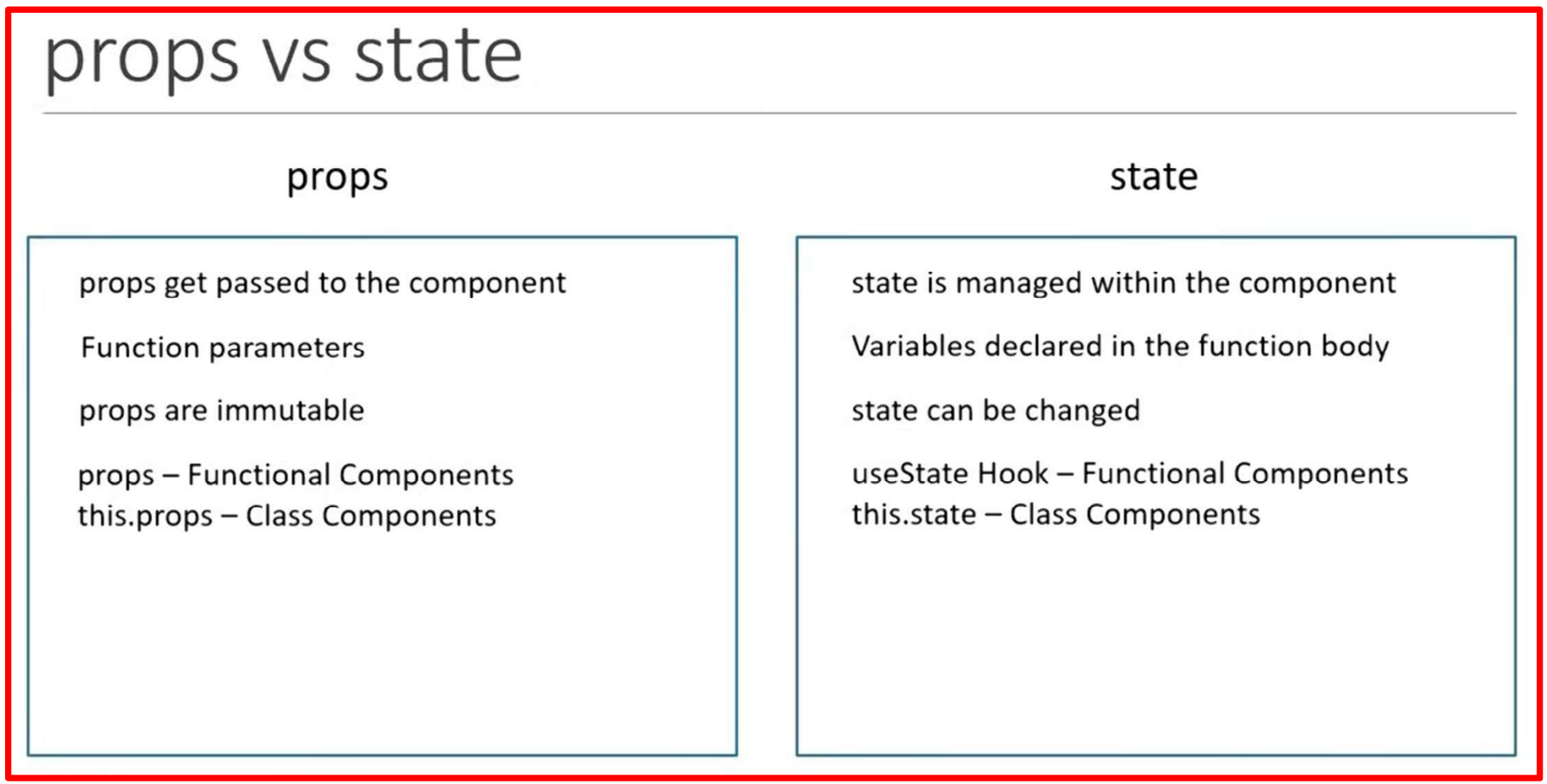


Introduction to State

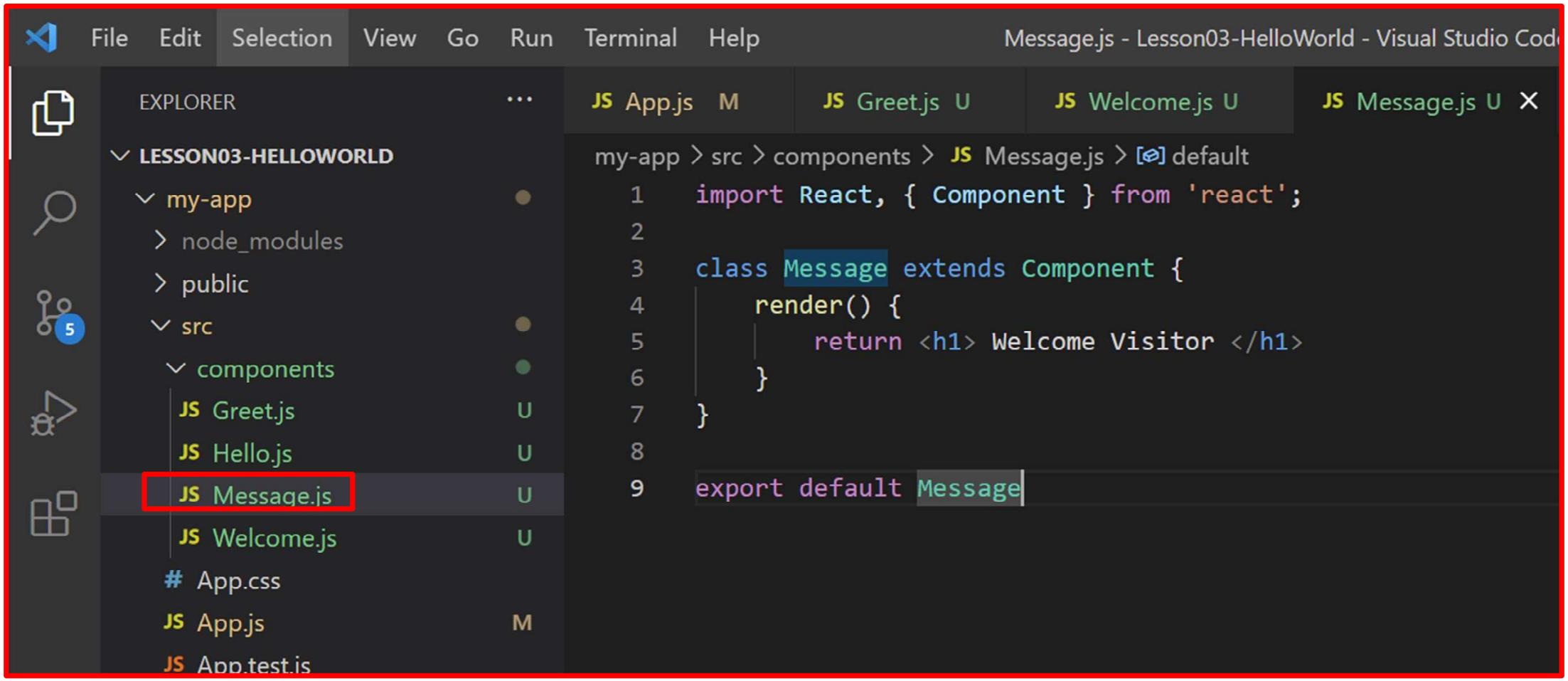
- As we have seen so far every react component returns JSX which describes the user interface.
- It is possible to influence what is rendered on the screen using props as it turns out
- There is a second way to influence what is rendered on the screen and that is the **State of the component.**

- **State in React components** is essential to manage and communicate data in your application.
- It is represented as a **JavaScript object** and has **component level scope**, it can be thought of as the **Private Data of Your Component**.

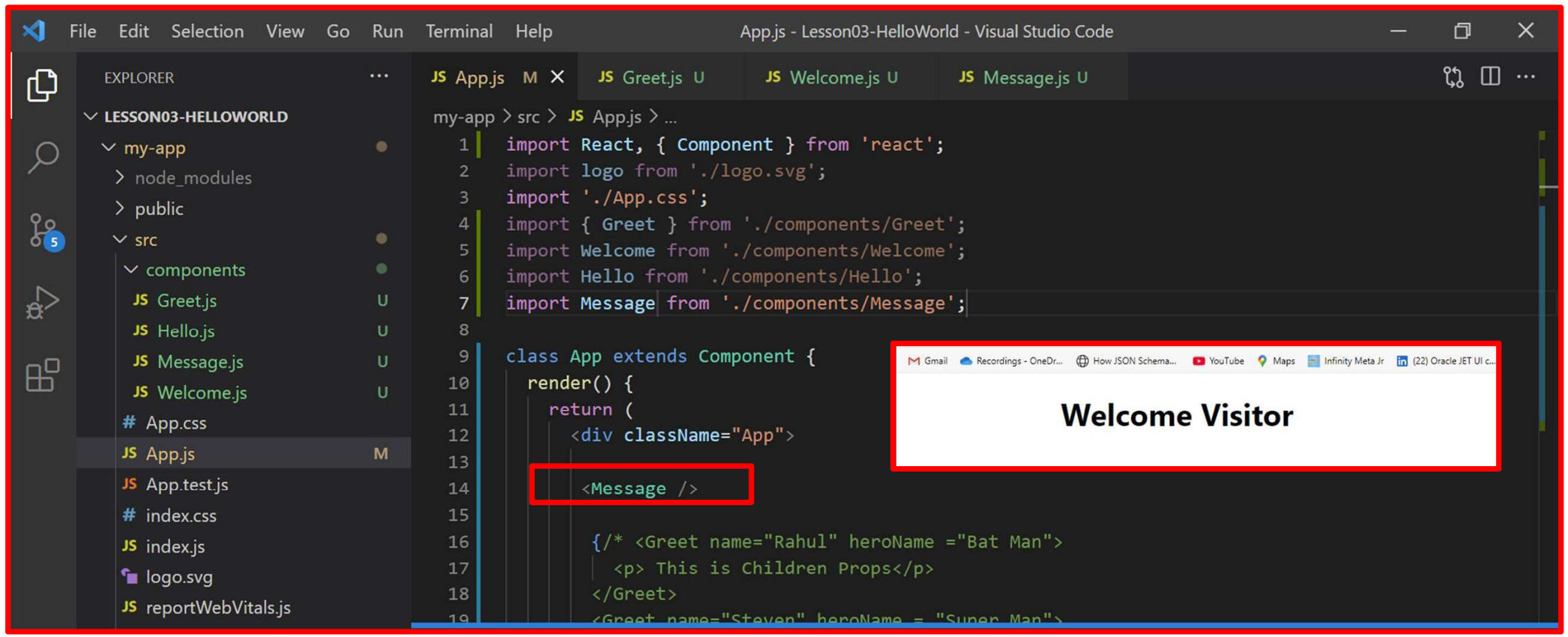
- Both **props** and **State** hold information that influences the UI in the browser



We shall Create Message.js



Including Message Component in App Component



Real Need ?

- Now here is our new requirement we need to have a subscribe button right below the text
- When we click on the button the text being displayed should change from **Welcome Visitor** to **Thank U for Subscribing**
- Now if the message were to be sent from App.js as a property it will not work as props are immutable once the message is said to Welcome Visitor
- It can never be changed from the message component and so the solution is to use
  - **Component State.**

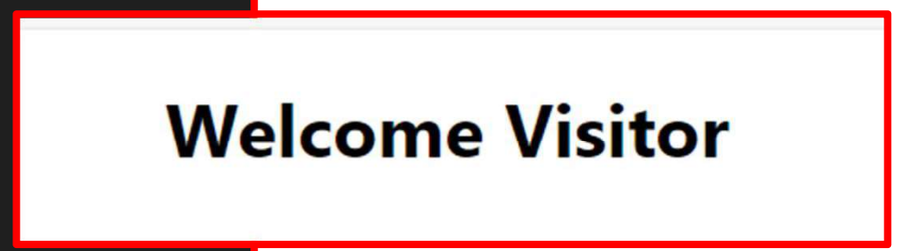
Steps Involved

1. Create a State Object and Initialize it. This is done usually in the Class Constructor

   - Now remember we are inside a class so don't forget to use the this keyword
   - State is reserved keyword in React
   - **[When we use this.state React Understands Our Intentions]**

Step 2

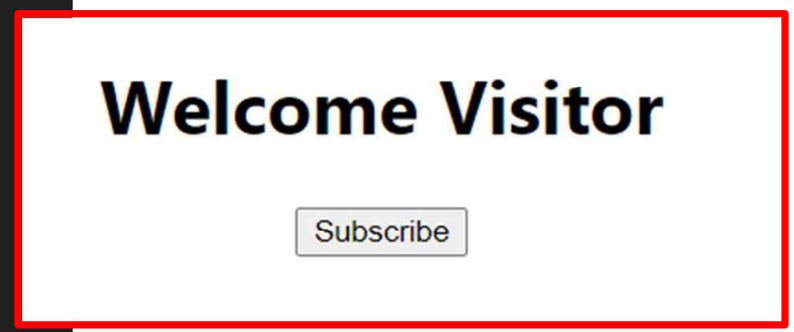
2. Bind this state value in the render function and we do this very similar to props instead of this.props we simply



Step 3 :

3. We are now using state to render the message we now have the ability to change this message

```
class Message extends Component {

    constructor() {
        super()
        this.state = {
            message: 'Welcome Visitor'
        }
    }

    render() {
        return (
            <div>
                <h1> { this.state.message } </h1>
                <button>Subscribe</button>
            </div>
        )
    }
}
```

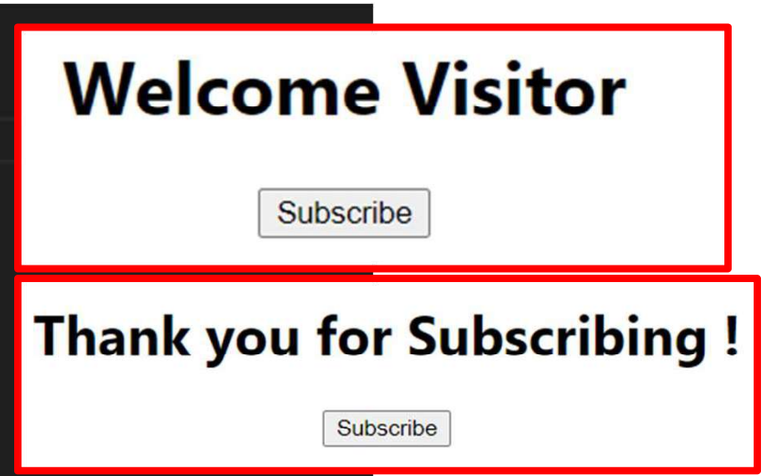


- We need an enclosing div tag because react wants only one element returned

Step 4 :

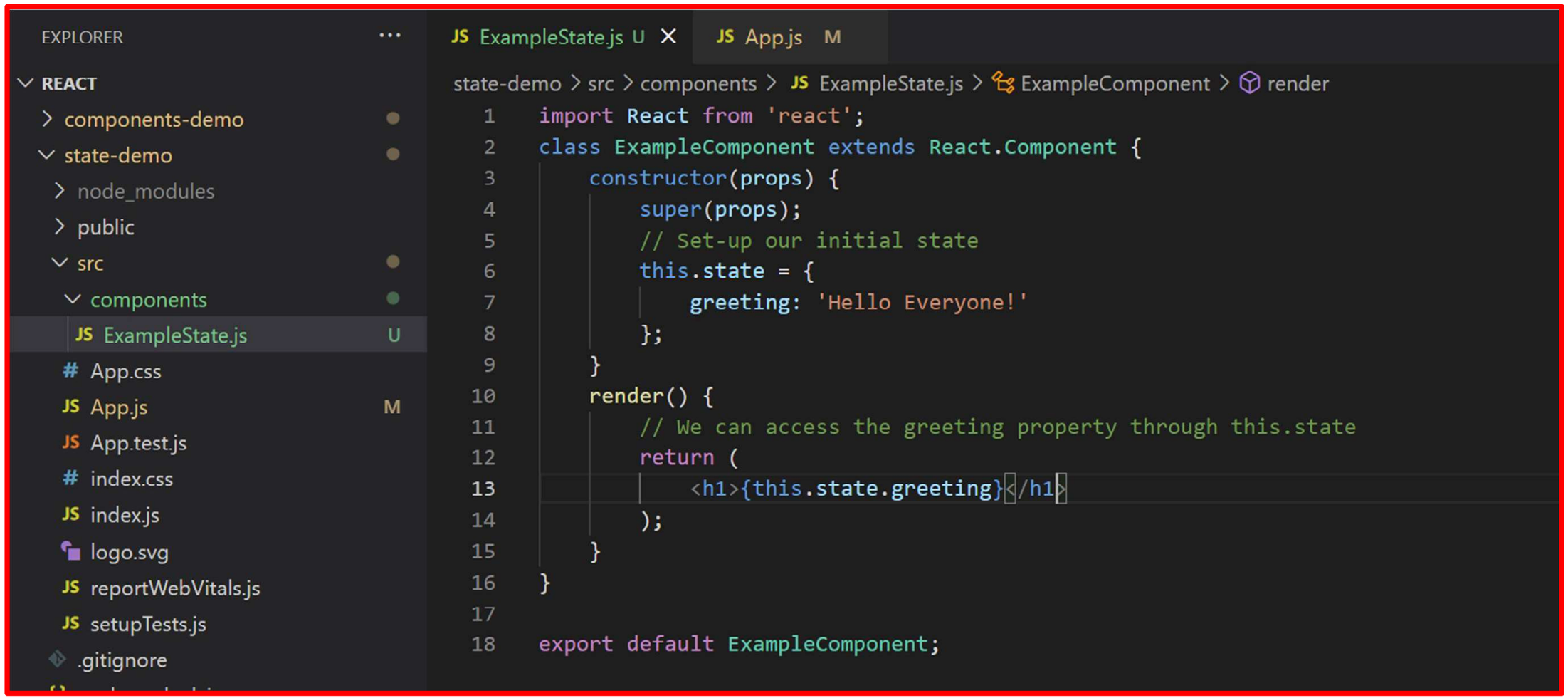
4. To listen to the click event on this button and change the message

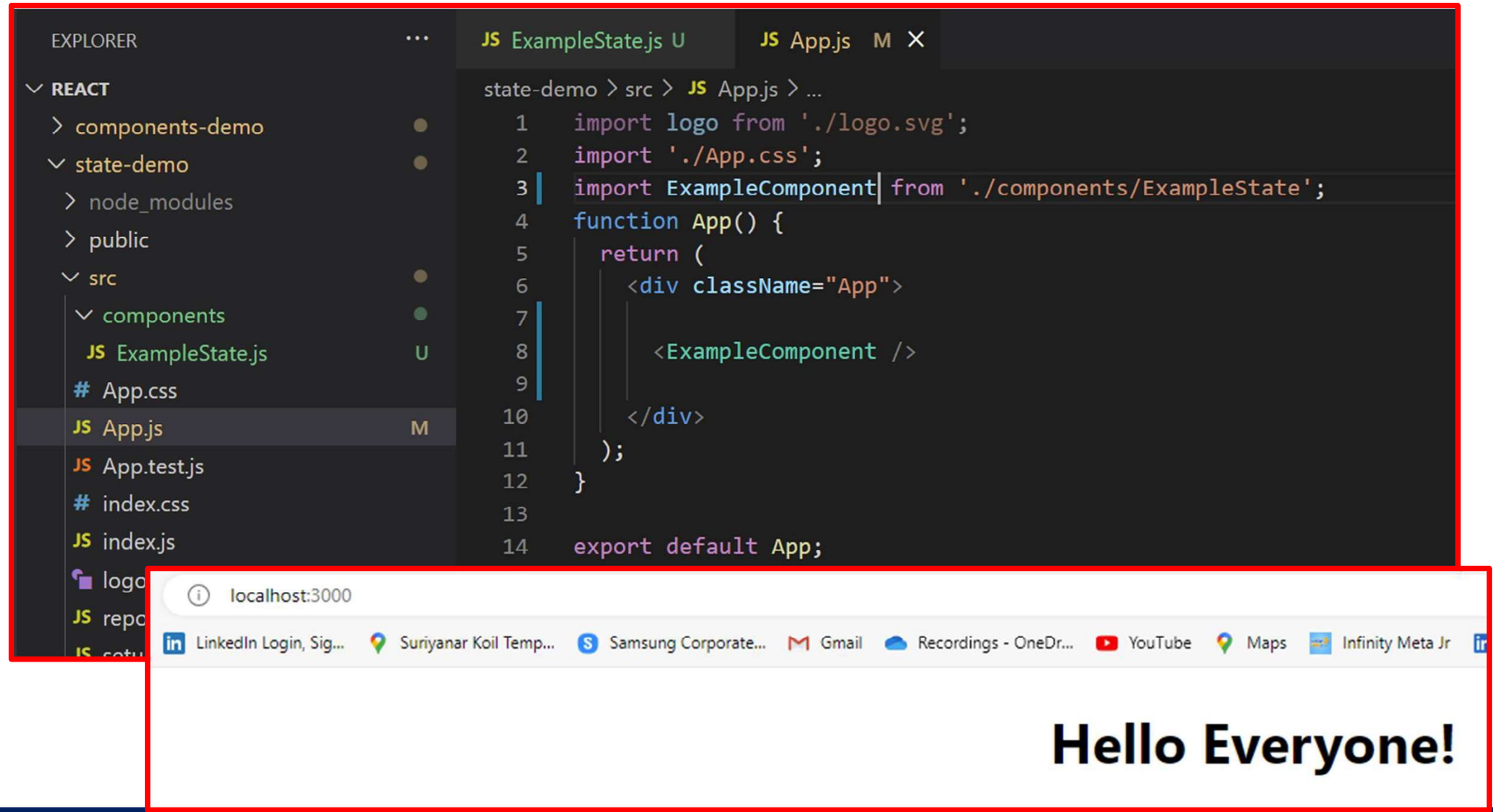
```
changeMessage() {
    this.setState({
        message: 'Thank you for Subscribing !'
    })
}

render() {
    return (
        <div>
            <h1> { this.state.message } </h1>
            <button onClick={ ()=> this.changeMessage() }>Subscribe</button>
        </div>
    )
}
```

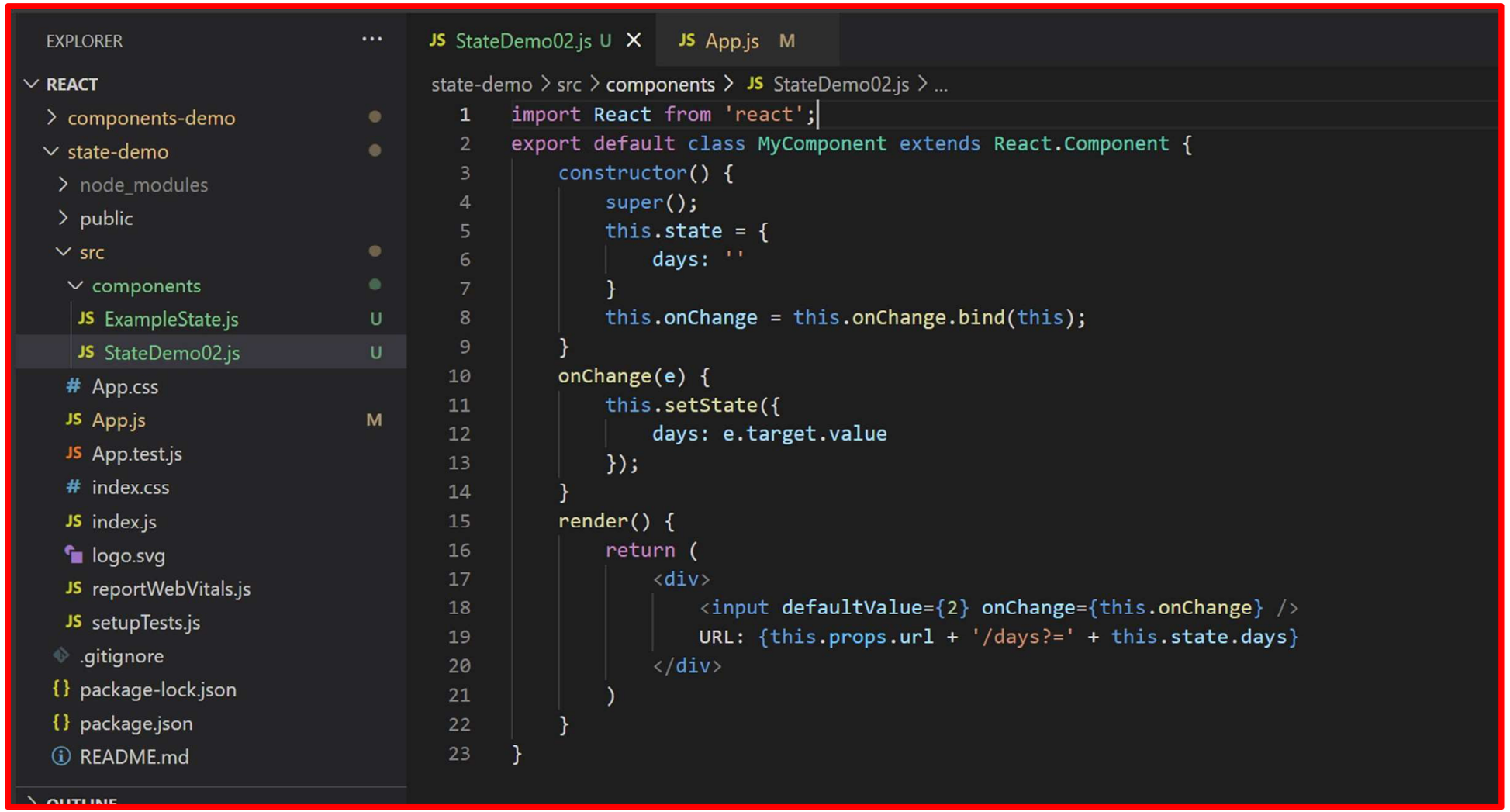


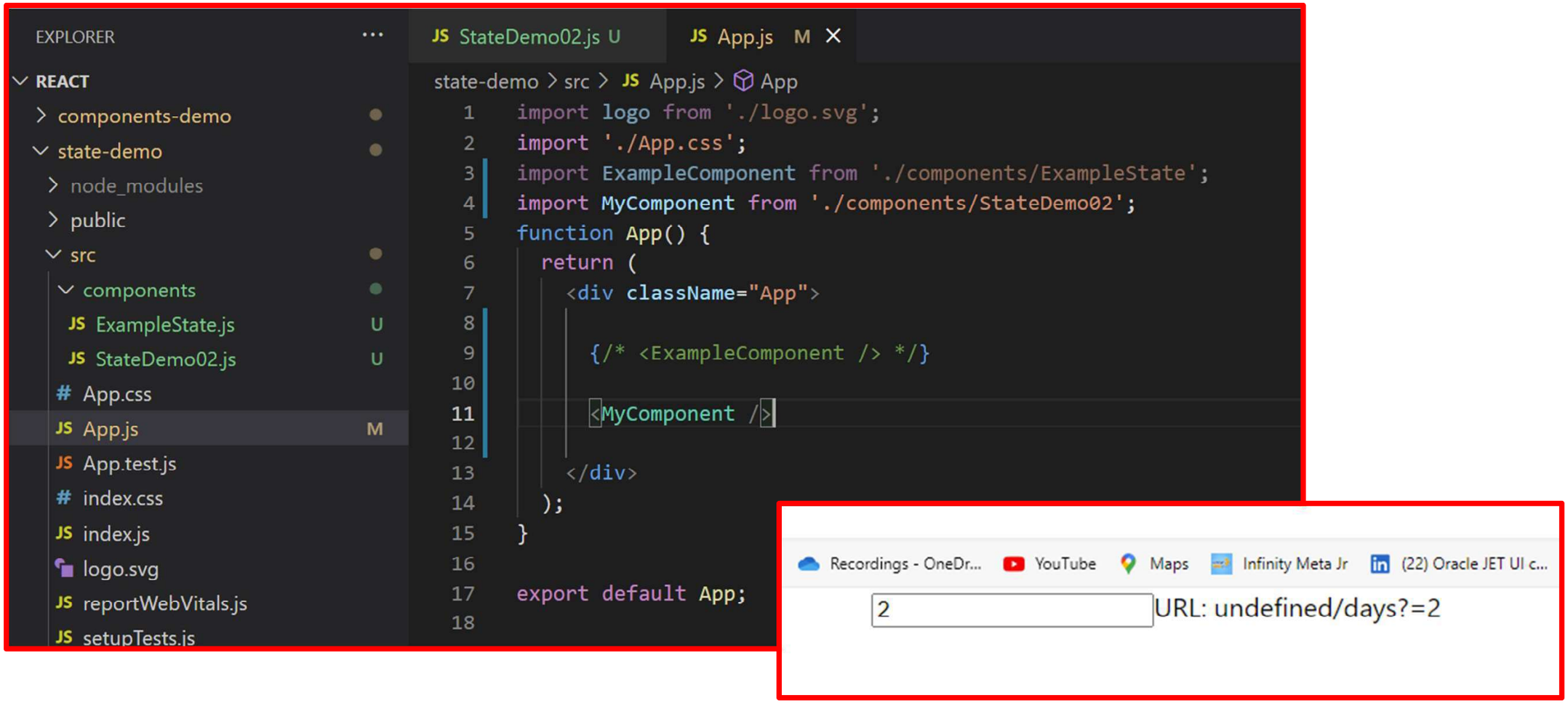
- **setState** method accepts an object which is nothing but the new state of the component in the new state

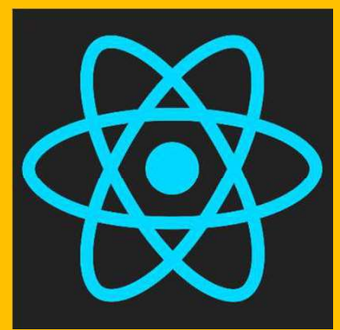
1. State in react is nothing but an object that is privately maintained inside a component.
2. State can influence what is rendered in the browser
3. Lastly state can be changed within the component **[Mutable]**


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
ExampleState.js U
App.js M
state-demo > src > components > ExampleState.js > ExampleComponent > render
1 import React from 'react';
2 class ExampleComponent extends React.Component {
3 constructor(props) {
4 super(props);
5 // Set-up our initial state
6 this.state = {
7 greeting: 'Hello Everyone!'
8 };
9 }
10 render() {
11 // We can access the greeting property through this.state
12 return (
13 <h1>{this.state.greeting}</h1>
14);
15 }
16 }
17
18 export default ExampleComponent;


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
App.css
App.js M
App.test.js
index.css
index.js
ExampleState.js U
App.js M
state-demo > src > App.js > ...
1 import logo from './logo.svg';
2 import './App.css';
3 import ExampleComponent from './components/ExampleState';
4 function App() {
5 return (
6
7
8 <ExampleComponent />
9
10
11);
12 }
13
14 export default App;
localhost:3000
LinkedIn Login, Sig...
Suriyanar Koil Temp...
Samsung Corporate...
Gmail
Recordings - OneDr...
YouTube
Maps
Infinity Meta Jr
Hello Everyone!


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
StateDemo02.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
StateDemo02.js U
App.js M
state-demo > src > components > StateDemo02.js > ...
1 import React from 'react';
2 export default class MyComponent extends React.Component {
3 constructor() {
4 super();
5 this.state = {
6 days: ''
7 }
8 this.onChange = this.onChange.bind(this);
9 }
10 onChange(e) {
11 this.setState({
12 days: e.target.value
13 });
14 }
15 render() {
16 return (
17
18 <input defaultValue={2} onChange={this.onChange} />
19 URL: {this.props.url + '/days?=' + this.state.days}
20
21)
22 }
23 }


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
StateDemo02.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
StateDemo02.js U
App.js M
state-demo > src > App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3 import ExampleComponent from './components/ExampleState';
4 import MyComponent from './components/StateDemo02';
5 function App() {
6 return (
7
8
9 {/* <ExampleComponent /> */}
10
11 <MyComponent />
12
13
14);
15 }
16
17 export default App;
18
Recordings - OneDr...
YouTube
Maps
Infinity Meta Jr
(22) Oracle JET UI c...
2
URL: undefined/days?=2



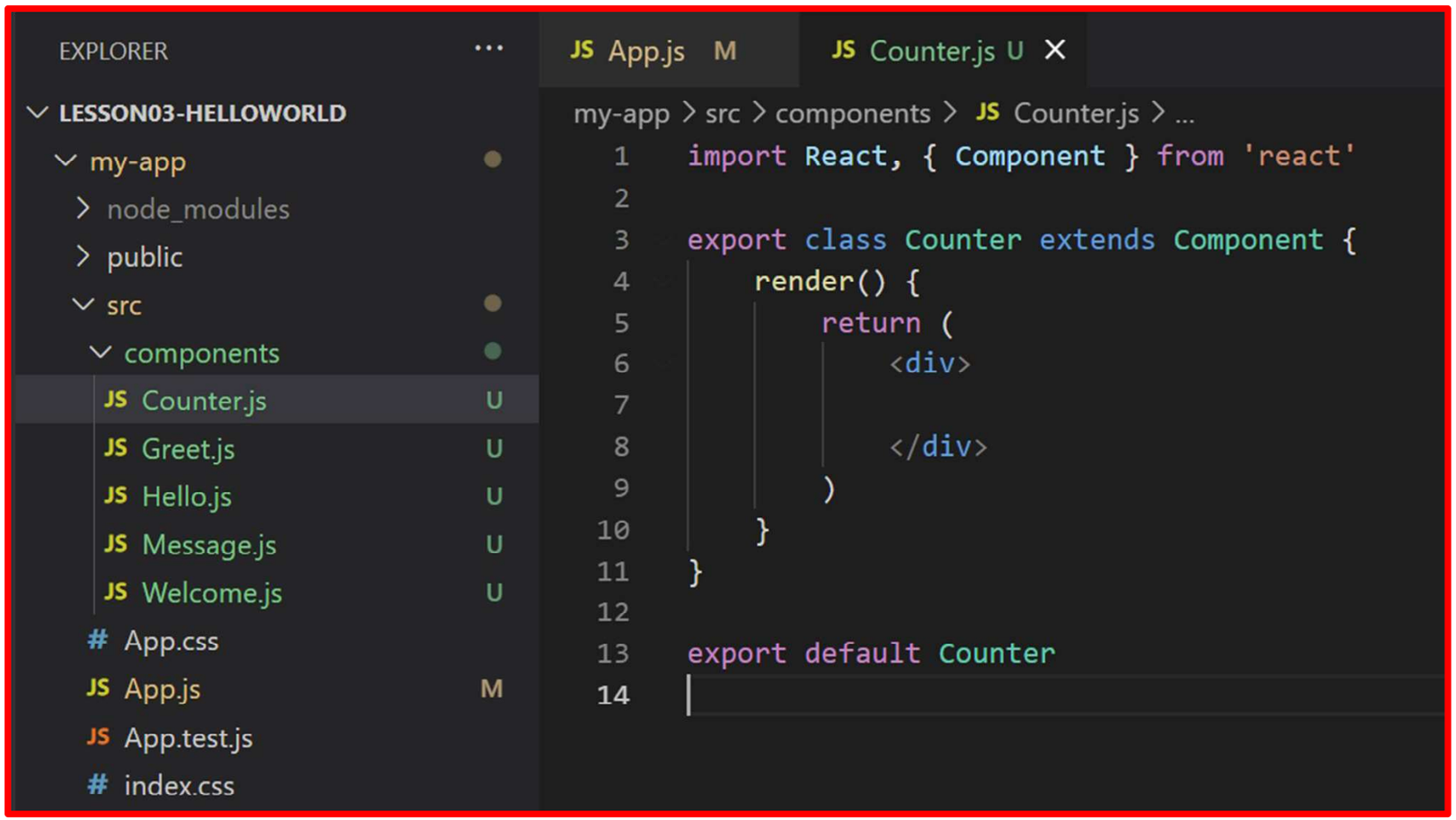
setState

setState()

- The primary way that you make UI updates to your React applications is through a call to the setState() function.
- This function will perform a *shallow merge* between the new state that you provide and the previous state, and will trigger a re-render of your component and all decedents

Counter Component

- Use Emmet Abbr [rce]



JS App.js M JS Counter.js U

my-app > src > components > JS Counter.js > default

```
1   import React, { Component } from 'react'
2
3   class Counter extends Component {
4       render() {
5           return (
6               <div>
7                   Count
8               </div>
9           )
10      }
11  }
12
13  export default Counter
14
```

JS App.js M JS Counter.js U

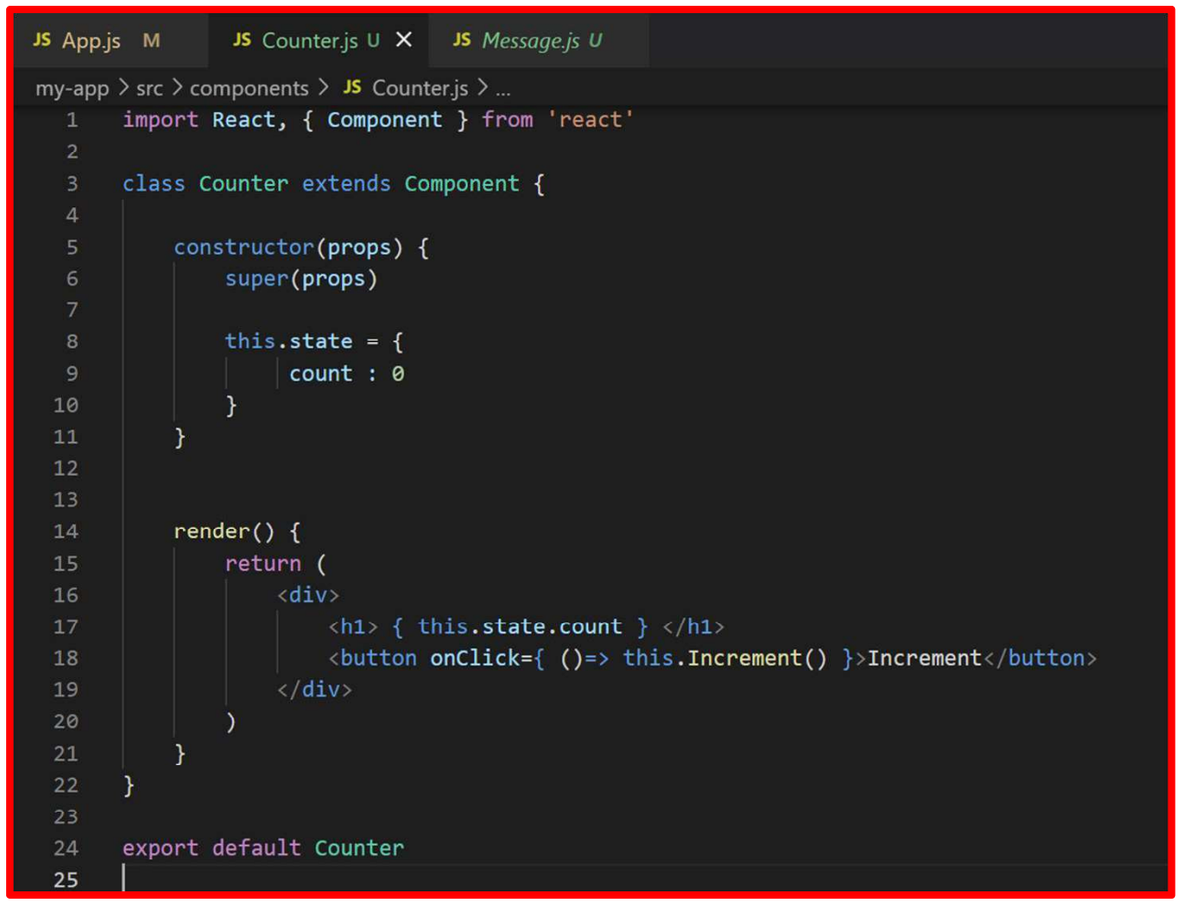
my-app > src > JS App.js > App > render

```
1   import React, { Component } from 'react';
2   import logo from './logo.svg';
3   import './App.css';
4   import { Greet } from './components/Greet';
5   import Welcome from './components/Welcome';
6   import Hello from './components/Hello';
7   import Message from './components/Message';
8   import Counter from './components/Counter';
9
10  class App extends Component {
11    render() {
12      return (
13        <div className="App">
14
15          <Counter />
```

dings - OneDr... How JSON Schema... YouTube Maps Infinity Meta

Count

- We need is a count state to keep track of the counter value
  - Use rconst in Emmet Abbr



What Happens if we Don't Use setState to Update and Directly Update it

```
increment() {
    this.state.count = this.state.count + 1
    console.log(this.state.count);
}
```

- This is the main reason Why we should not modify State Directly
- The only place where you can assign this.state is the constructor any other time to change the state set state method has to be used

```
increment() {
    this.setState({
        count: this.state.count + 1
    })
    console.log(this.state.count);
}
```

Parameters

1. updater: It can be an object with a number of key-value pairs that should be merged into the state or a function that returns such an object.

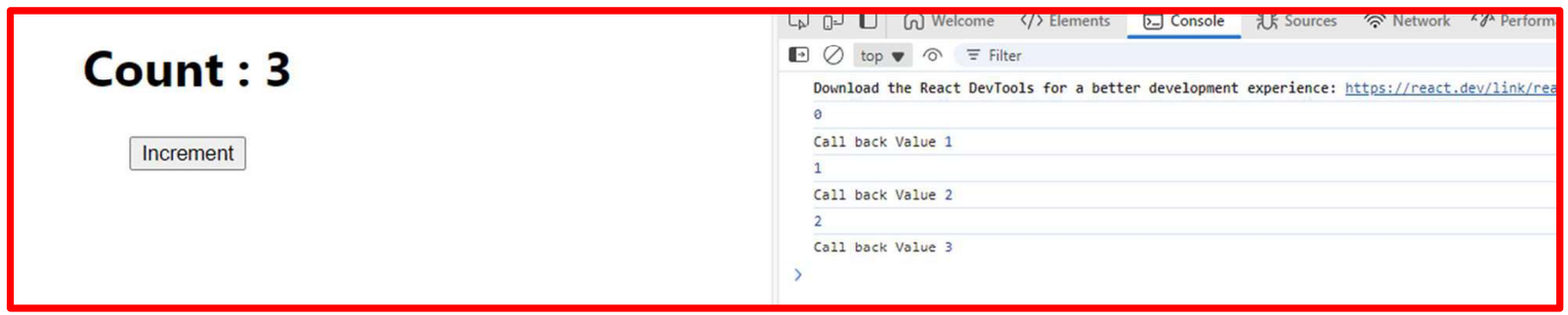
2. callback (optional): a function which will be executed after setState() has been executed successfully.

Due to the fact that calls to setState() are not guaranteed by React to be atomic, this can sometimes be useful if you want to perform some action after you are positive that setState() has been executed successfully.

Observe ….

- There is one detail to observe with said state in the browser you can see that when we click on increment the value is 1 but if you take a look at the console the value is 0
- So the console value is 1 less than the rendered value and this is because calls to set state are asynchronous so what is happening is console.log over here is being called before the state is actually set
- Many a times in your application you might want to execute some code only after the state has been updated to handle such a situation you can pass in a callback function as the second parameter to the set state method so the set state method has two parameters the **first parameter** is the **state object** and **the second parameter** now is the **callback function** the callback function will be an arrow function

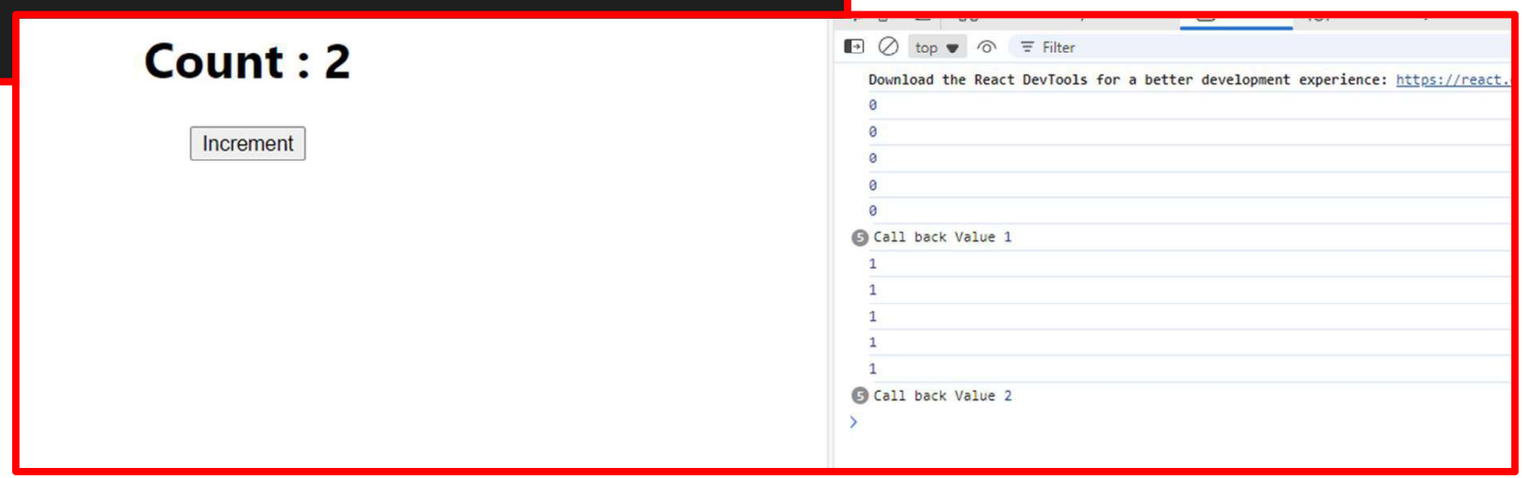
```
increment() {
    this.setState({
        count: this.state.count + 1
    },
    () => {
            console.log('Call back Value', this.state.count)
        }
    )
    console.log(this.state.count);
}
```



- whenever you need to execute some code after the state has been changed do not place that code right after the said state method instead place that code within the callback function that is passed as a second parameter to the said state method.

Another UseCase

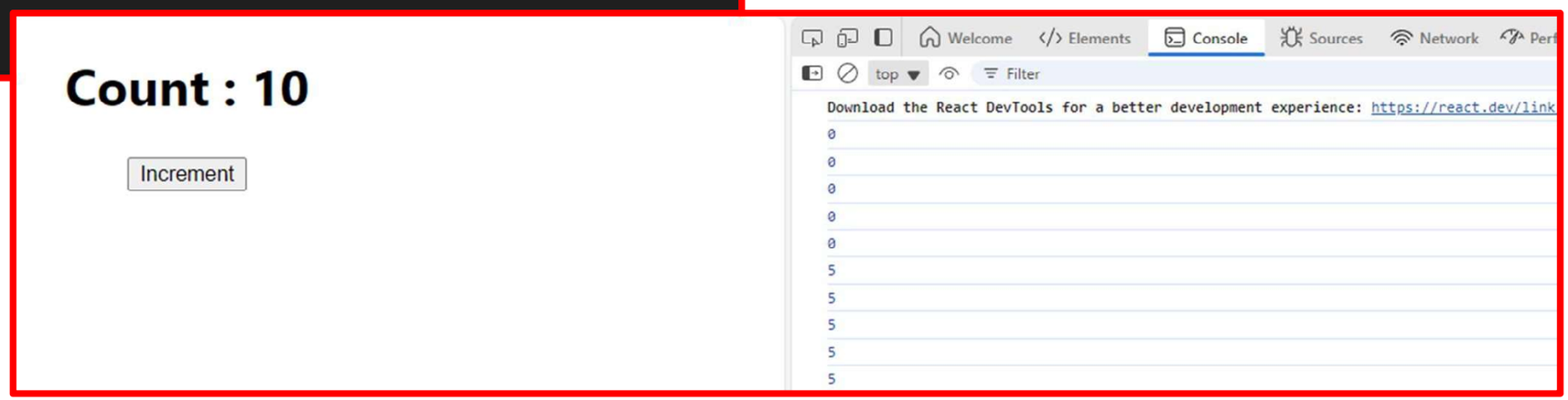
```
23
24      incrementFive() {
25          this.increment()
26          this.increment()
27          this.increment()
28          this.increment()
29          this.increment()
30      }
31      render() {
32          return (
33              <div>
34                  <h1> Count : {this.state.count} </h1>
35                  <button onClick={() => this.incrementFive()}>Increment</button>
36              </div>
37          )
38      }
```



Very Important Point to Note

- when we click on the button you can see that the value changes to one instead of changing to five and in the console strangely zero is logged five times and even the callback value of one is logged five times.
- Now why is this behavior is because react may group multiple set state calls into a single update for better performance so what happens in our scenario is that all the five set state calls are done in one single go and the updated value does not carry over between the different calls so whenever you have to update the state based on the previous state we need to pass a function as an argument to set state method instead of passing in an object so this dot said state and as the argument we pass in a function

```
increment() {
    // this.setState({
    //     count: this.state.count + 1
    // },
    // () => {
    //         console.log('Call back Value', this.state.count)
    //     }
    // )
    this.setState(prevState => ({
        count: prevState.count + 1
    }))
    console.log(this.state.count);
}
```



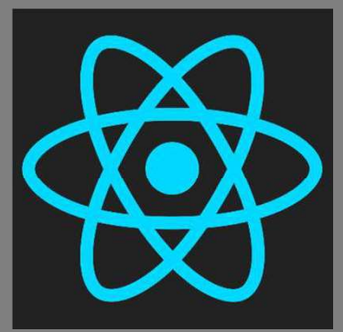
- When you have to update the state based on the previous state make sure to pass in a function as an argument instead of the regular object the function has access to the previous state which can be used to calculate the new state

setState

Always make use of setState and never modify the state directly.

Code has to be executed after the state has been updated ? Place that code in the call back function which is the second argument to the setState method.

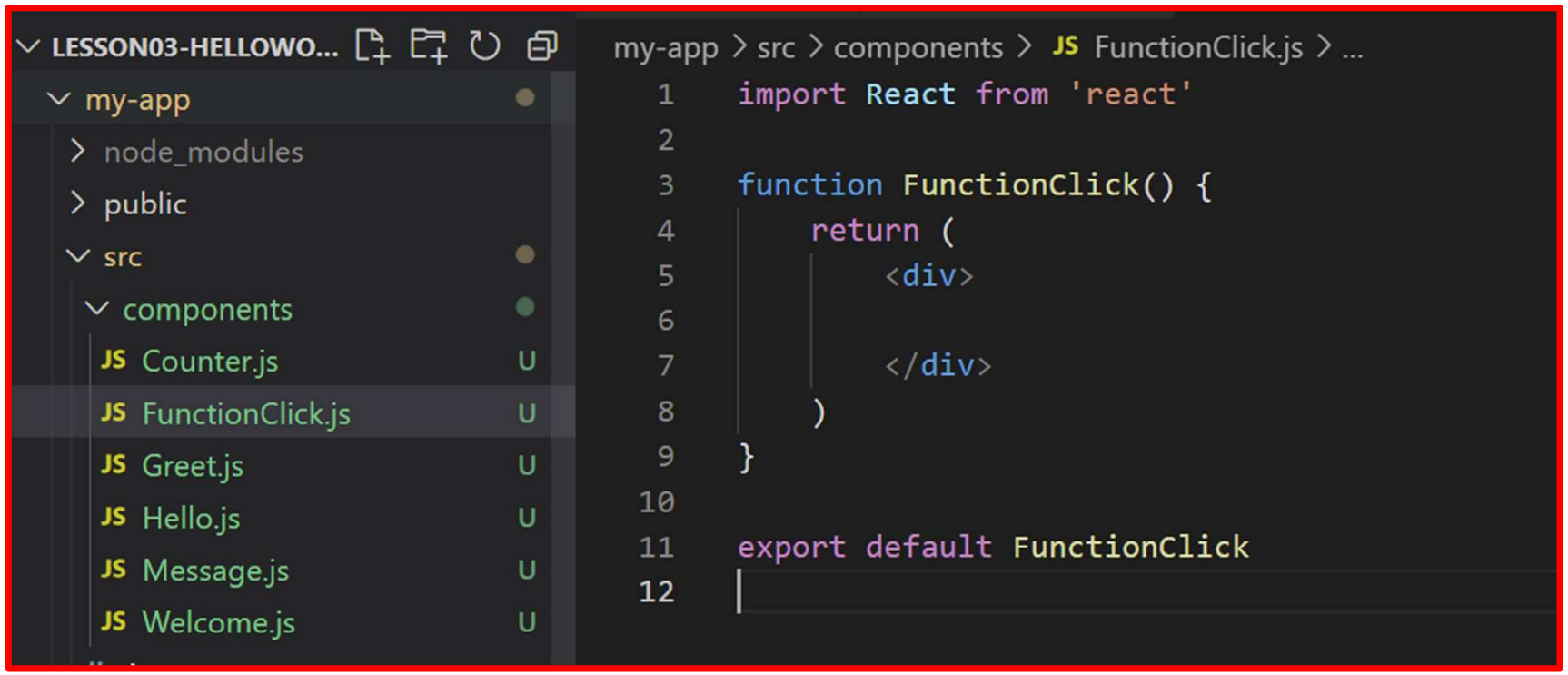
When you have to update state based on the previous state value, pass in a function as an argument instead of the regular object.



Event Handling

- Any web application you create typically tends to have user interaction when the user interacts with your application events are fired
  - Eg: mouse clicks, mouse over, key press change event and so on the application must handle such events and execute the necessary code

- Create a Functional Component [rfce is Emmet Abbr]



JS App.js M JS Counter.js U JS FunctionClick.js

my-app > src > components > JS FunctionClick.js > ...

```
1  import React from 'react'
2
3  function FunctionClick() {
4      return (
5          <div>
6              <button>Click</button>
7          </div>
8      )
9  }
10
11 export default FunctionClick
12
```

JS App.js M JS Counter.js U JS FunctionClick.js U JS *Message.js U*

my-app > src > JS App.js > App > render

```
1  import React, { Component } from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4  import { Greet } from './components/Greet';
5  import Welcome from './components/Welcome';
6  import Hello from './components/Hello';
7  import Message from './components/Message';
8  import Counter from './components/Counter';
9  import FunctionClick from './components/FunctionClick';
10 class App extends Component {
11   render() {
12     return (
13       <div className="App">
14
15         <FunctionClick />
16         {/* <Counter /> */}
```

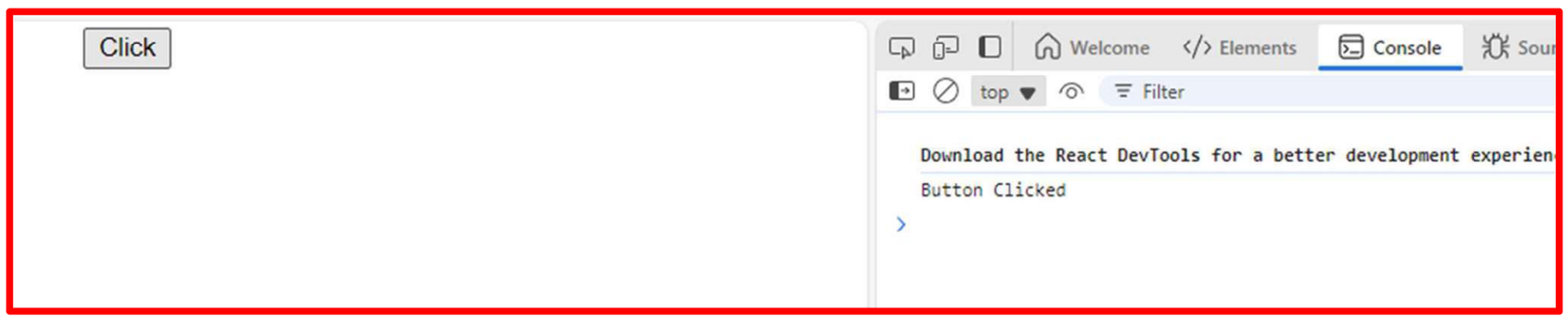
ings - OneDr... How JSON Schema... YouTube Maps

Click

App.js M FunctionClick.js U

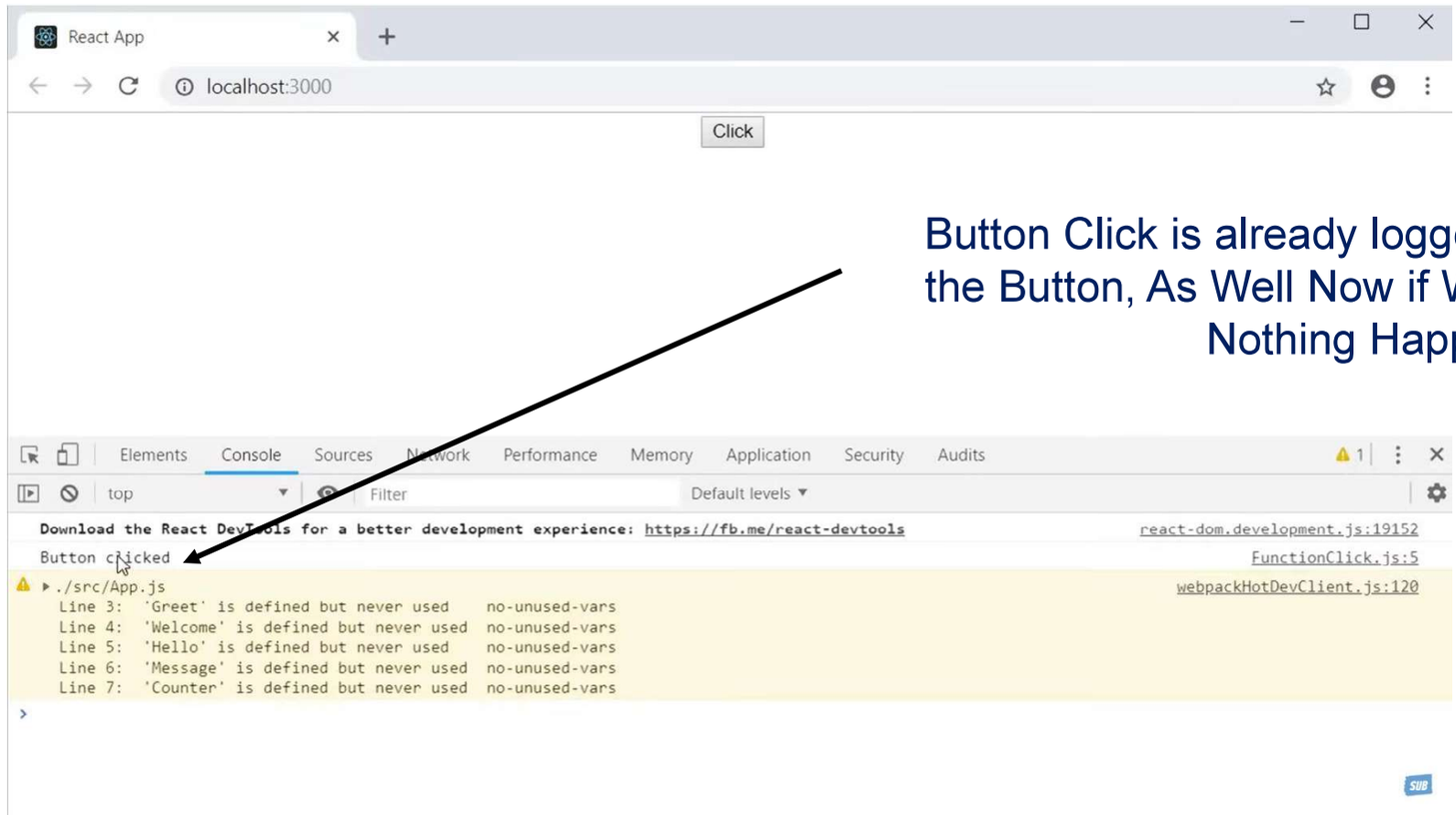
my-app > src > components > FunctionClick.js > FunctionClick > clickHandler

```
1   import React from 'react'
2
3   function FunctionClick() {
4
5       function clickHandler() {
6           console.log('Button Clicked ')
7       }
8       return (
9           <div>
10              <button onClick= { clickHandler }>Click</button>
11          </div>
12      )
13  }
14
15  export default FunctionClick
16
```

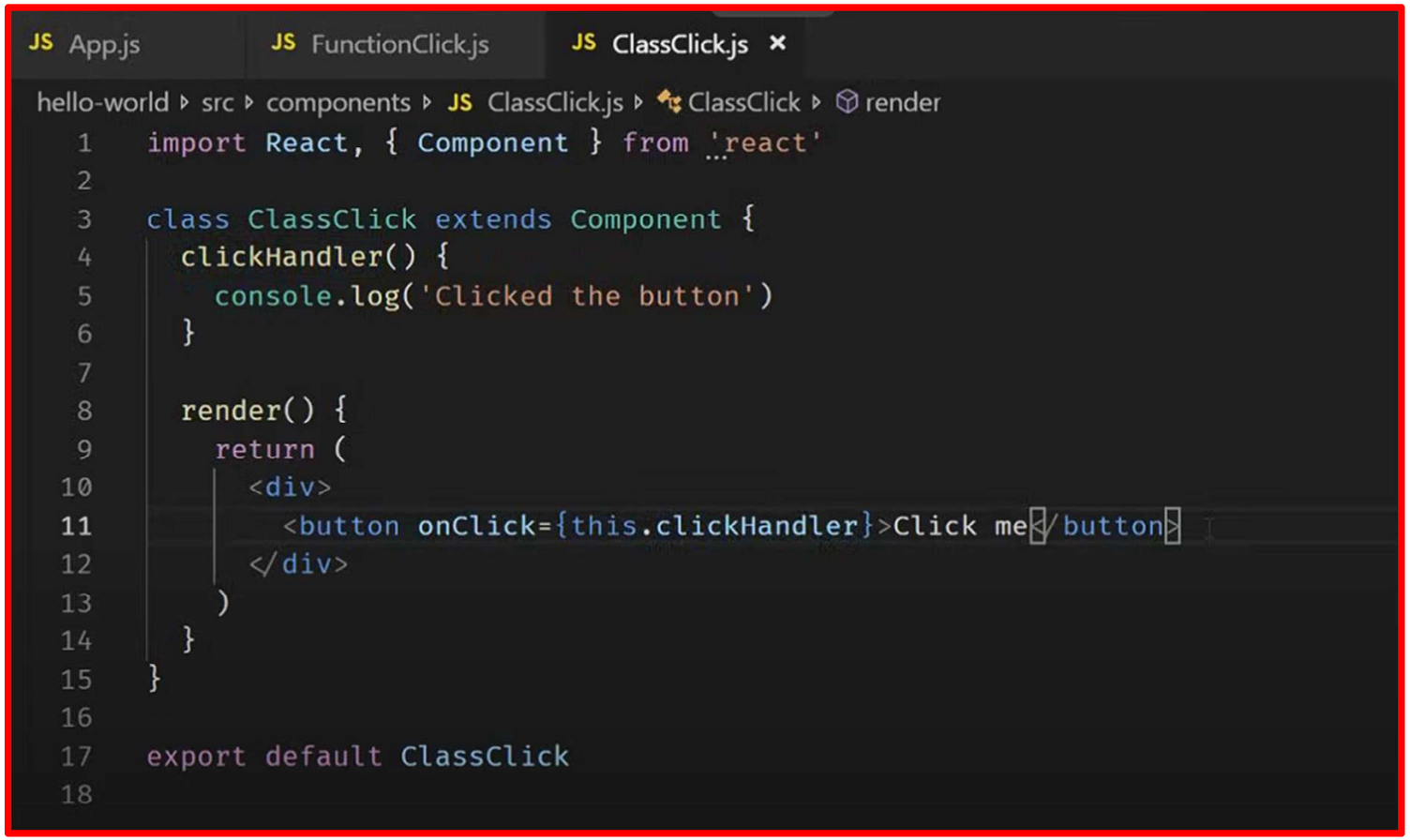


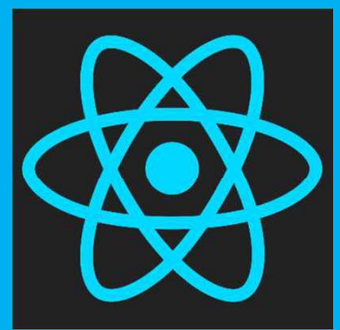
Never Use () in the Handler Function

- If you do add parentheses it becomes a function call and that is not what we want we want the handler to be a function and not a function call.



Event's in Class Component

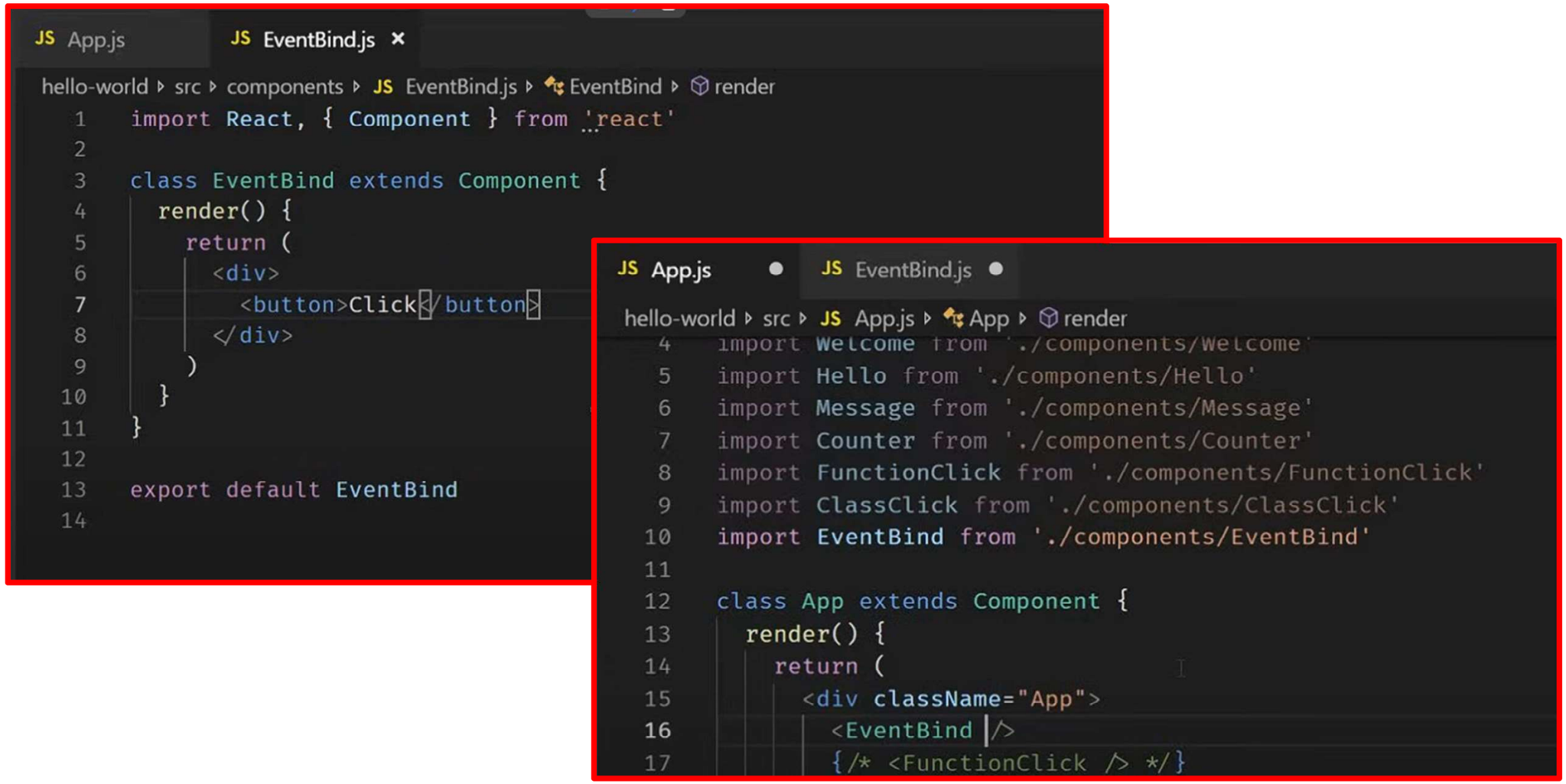




Event Binding

Event Binding

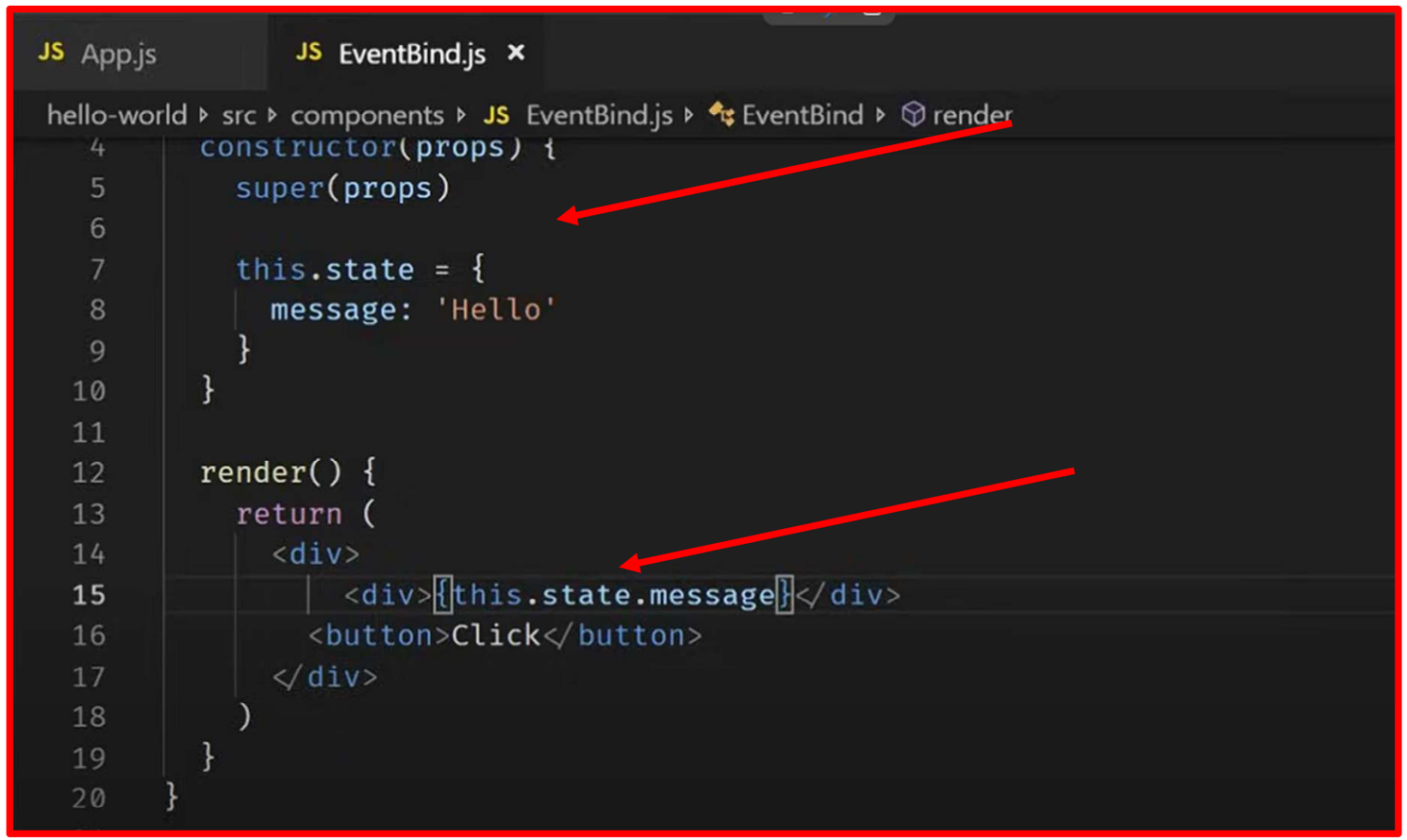
- How to bind event handlers in react components ?
  - The reason we bind event handlers in react purely because of the way this keyword works in JavaScript it is not because of how react works


App.js
EventBind.js
hello-world ▸ src ▸ components ▸ EventBind.js ▸ EventBind ▸ render
import React, { Component } from 'react'
class EventBind extends Component {
render() {
return (

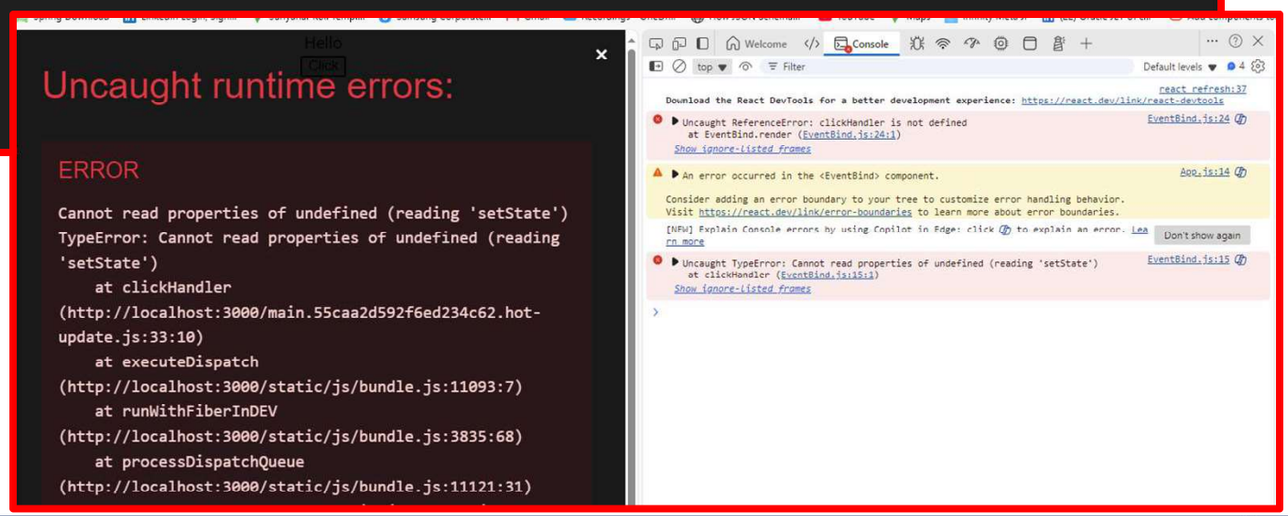
<button>Click</button>

)
}
}
export default EventBind
App.js
EventBind.js
hello-world ▸ src ▸ App.js ▸ App ▸ render
import Welcome from './components/Welcome'
import Hello from './components/Hello'
import Message from './components/Message'
import Counter from './components/Counter'
import FunctionClick from './components/FunctionClick'
import ClassClick from './components/ClassClick'
import EventBind from './components/EventBind'
class App extends Component {
render() {
return (

<EventBind />
{/* <FunctionClick /> */}


JS App.js
JS EventBind.js
hello-world ▸ src ▸ components ▸ JS EventBind.js ▸ EventBind ▸ render
4 constructor(props) {
5 super(props)
6
7 this.state = {
8 message: 'Hello'
9 }
10 }
11
12 render() {
13 return (
14
15 {this.state.message}
16 <button>Click</button>
17
18)
19 }
20 }

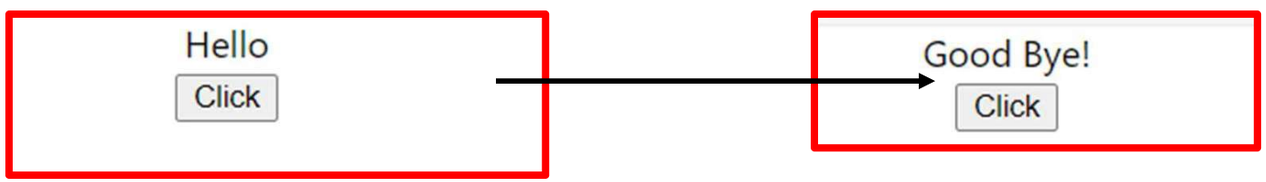
```
14      clickHandler() {
15          this.setState({
16              message : 'Good Bye!'
17          })
18      }
19
20      render() {
21          return (
22              <div>
23                  <div> { this.state.message }</div>
24                  <button onClick= { this.clickHandler }>Click</button>
25              </div>
26          )
27      }
28  }
```

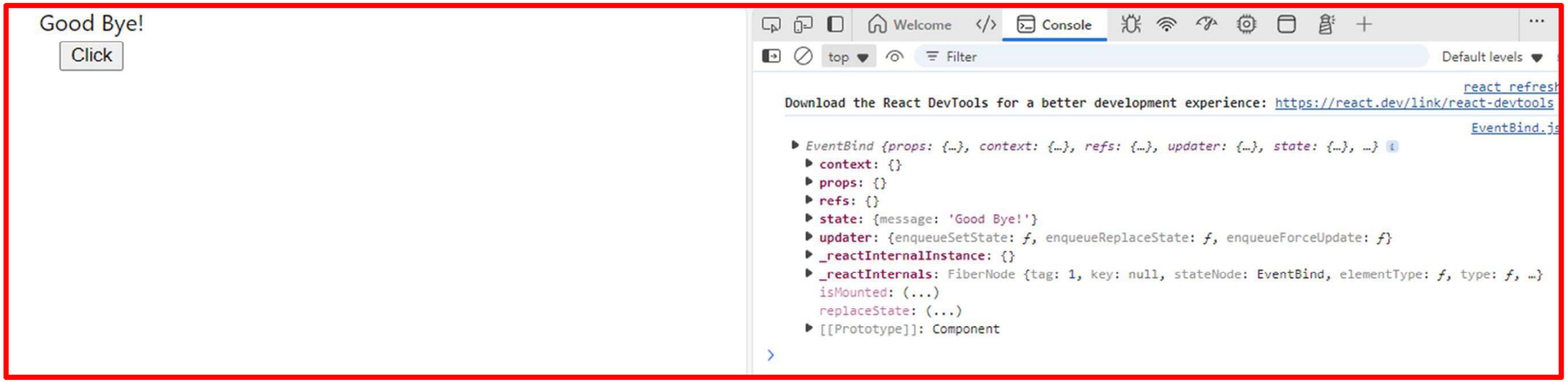


Different variants of binding the Event Handler

- The first option we have is to use the bind keyword and bind the handler in the render method

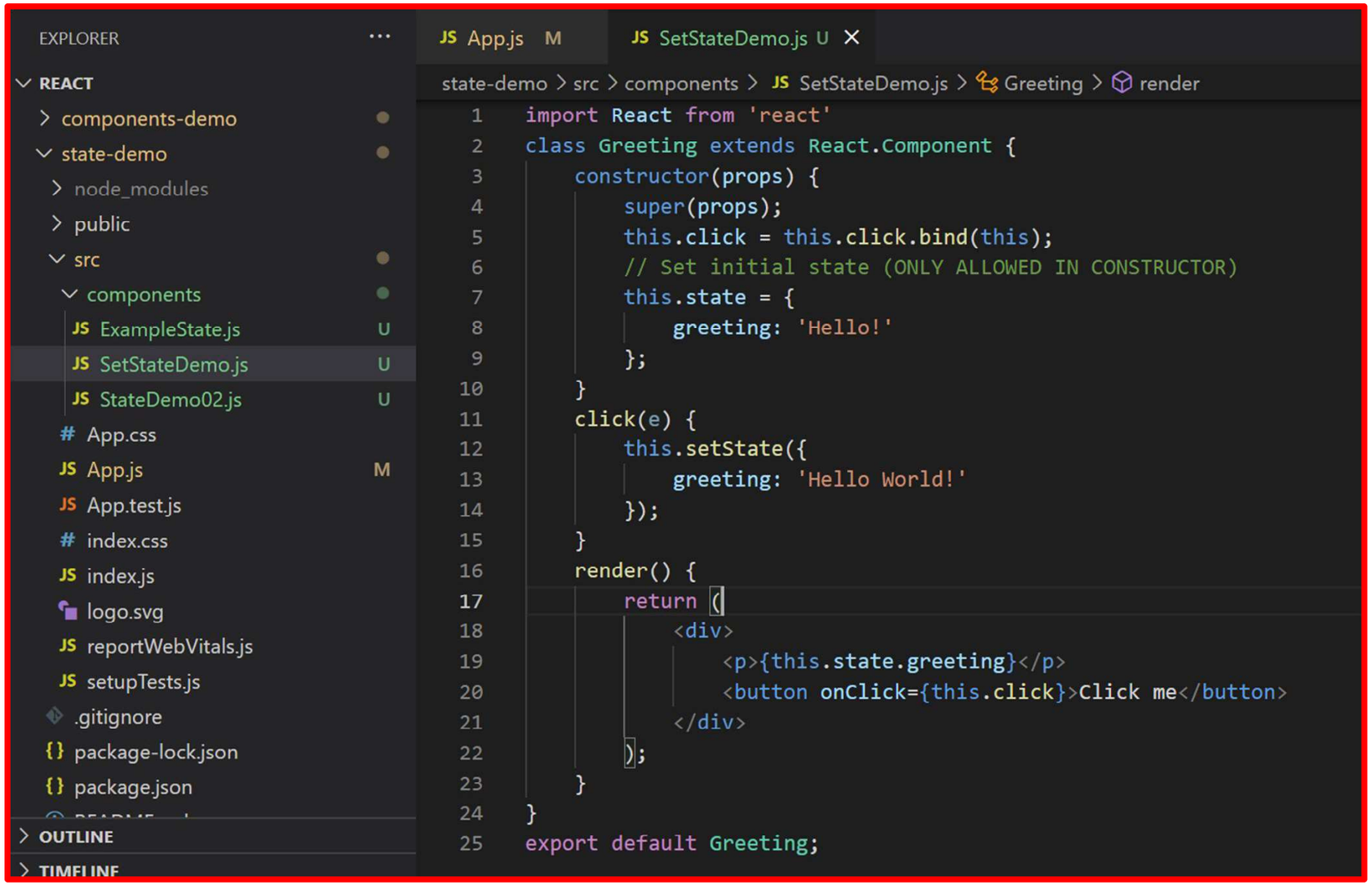
```
clickHandler() {
    this.setState({
        message : 'Good Bye!'
    })
    console.log(this)
}

render() {
    return (
        <div>
            <div> { this.state.message }</div>
            <button onClick= { this.clickHandler.bind(this) }>Click</button>
        </div>
    )
}
```




Good Bye!
Click
Welcome
Console
top
Filter
Default levels
react_refresh
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools
EventBind.js
EventBind {props: {…}, context: {…}, refs: {…}, updater: {…}, state: {…}, …}
context: {}
props: {}
refs: {}
state: {message: 'Good Bye!'}
updater: {enqueueSetState: ƒ, enqueueReplaceState: ƒ, enqueueForceUpdate: ƒ}
_reactInternalInstance: {}
_reactInternals: FiberNode {tag: 1, key: null, stateNode: EventBind, elementType: ƒ, type: ƒ, …}
isMounted: (...)
replaceState: (...)
[[Prototype]]: Component

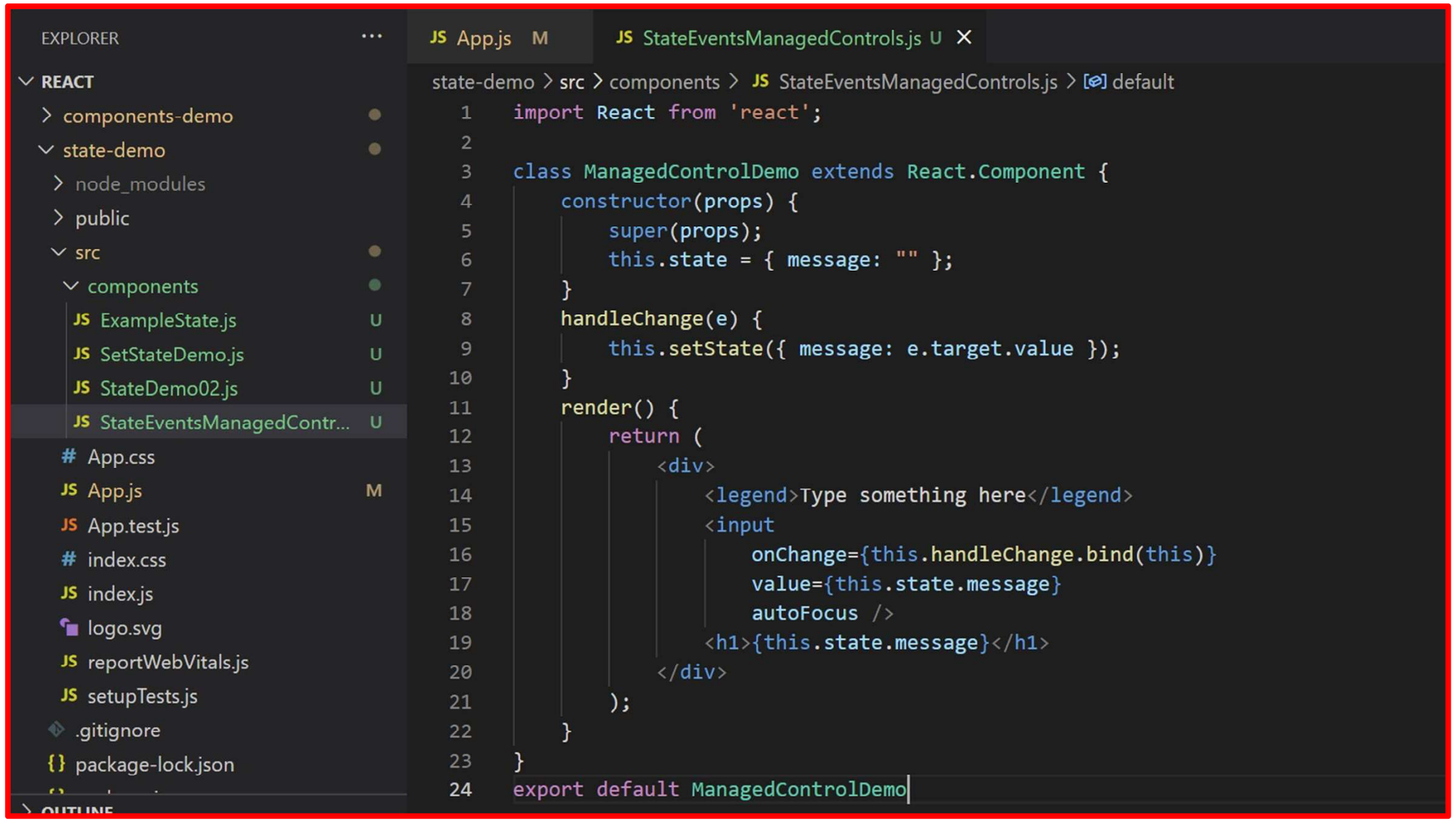
Second Approach for Binding Events [Using Arrow Function]

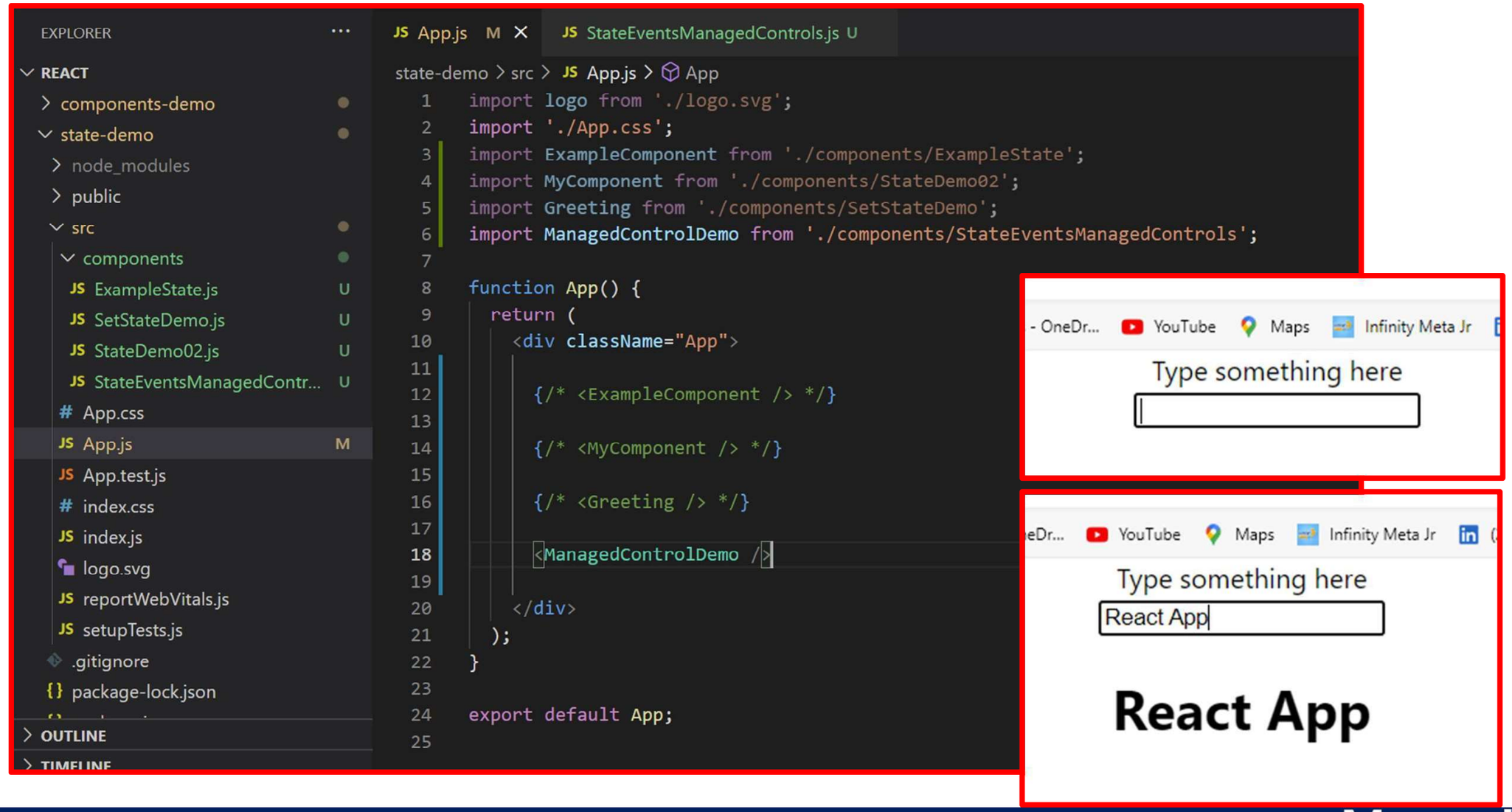
- It could be troublesome in large applications and components that contain nested children components so approach one binding in the render method the second approach is to use arrow functions in the render method


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
SetStateDemo.js U
StateDemo02.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
OUTLINE
TIMELINE
App.js M
SetStateDemo.js U
state-demo > src > components > SetStateDemo.js > Greeting > render
1 import React from 'react'
2 class Greeting extends React.Component {
3 constructor(props) {
4 super(props);
5 this.click = this.click.bind(this);
6 // Set initial state (ONLY ALLOWED IN CONSTRUCTOR)
7 this.state = {
8 greeting: 'Hello!'
9 };
10 }
11 click(e) {
12 this.setState({
13 greeting: 'Hello World!'
14 });
15 }
16 render() {
17 return (
18
19 {this.state.greeting}
20 <button onClick={this.click}>Click me</button>
21
22);
23 }
24 }
25 export default Greeting;

State, Events And Managed Controls

- A React component with a "managed" input field. Whenever the value of the input field
- changes, an event handler is called which updates the state of the component with the new value of the input field.
- The call to setState in the event handler will trigger a call to render updating the component in the dom.


EXPLORER
App.js M
StateEventsManagedControls.js U
state-demo > src > components > StateEventsManagedControls.js > default
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
SetStateDemo.js U
StateDemo02.js U
StateEventsManagedContr... U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
OUTLINE
1 import React from 'react';
2
3 class ManagedControlDemo extends React.Component {
4 constructor(props) {
5 super(props);
6 this.state = { message: "" };
7 }
8 handleChange(e) {
9 this.setState({ message: e.target.value });
10 }
11 render() {
12 return (
13
14 <legend>Type something here</legend>
15 <input
16 onChange={this.handleChange.bind(this)}
17 value={this.state.message}
18 autoFocus />
19 <h1>{this.state.message}</h1>
20
21);
22 }
23 }
24 export default ManagedControlDemo


EXPLORER
REACT
components-demo
state-demo
node_modules
public
src
components
ExampleState.js U
SetStateDemo.js U
StateDemo02.js U
StateEventsManagedContr... U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
OUTLINE
TIMELINE
App.js M
StateEventsManagedControls.js U
state-demo > src > App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3 import ExampleComponent from './components/ExampleState';
4 import MyComponent from './components/StateDemo02';
5 import Greeting from './components/SetStateDemo';
6 import ManagedControlDemo from './components/StateEventsManagedControls';
7
8 function App() {
9 return (
10
11
12 {/* <ExampleComponent /> */}
13
14 {/* <MyComponent /> */}
15
16 {/* <Greeting /> */}
17
18 <ManagedControlDemo />
19
20
21);
22 }
23
24 export default App;
25
- OneDr... YouTube Maps Infinity Meta Jr
Type something here
eDr... YouTube Maps Infinity Meta Jr
Type something here
React App
React App

- Its very important to note the runtime behavior. Every time a user changes the value in the input field
  1. handleChange will be called and so
  2. setState will be called and so
  3. render will be called

Summary

In this lesson, you should have learned how to:

- Basics in State
- setState
- State, Events and Managed Controls