

props vs state

props

props get passed to the component

Function parameters

props are immutable

props – Functional Components

this.props – Class Components

state

state is managed within the component

Variables declared in the function body

state can be changed

useState Hook – Functional Components

this.state – Class Components

What is State

State Example

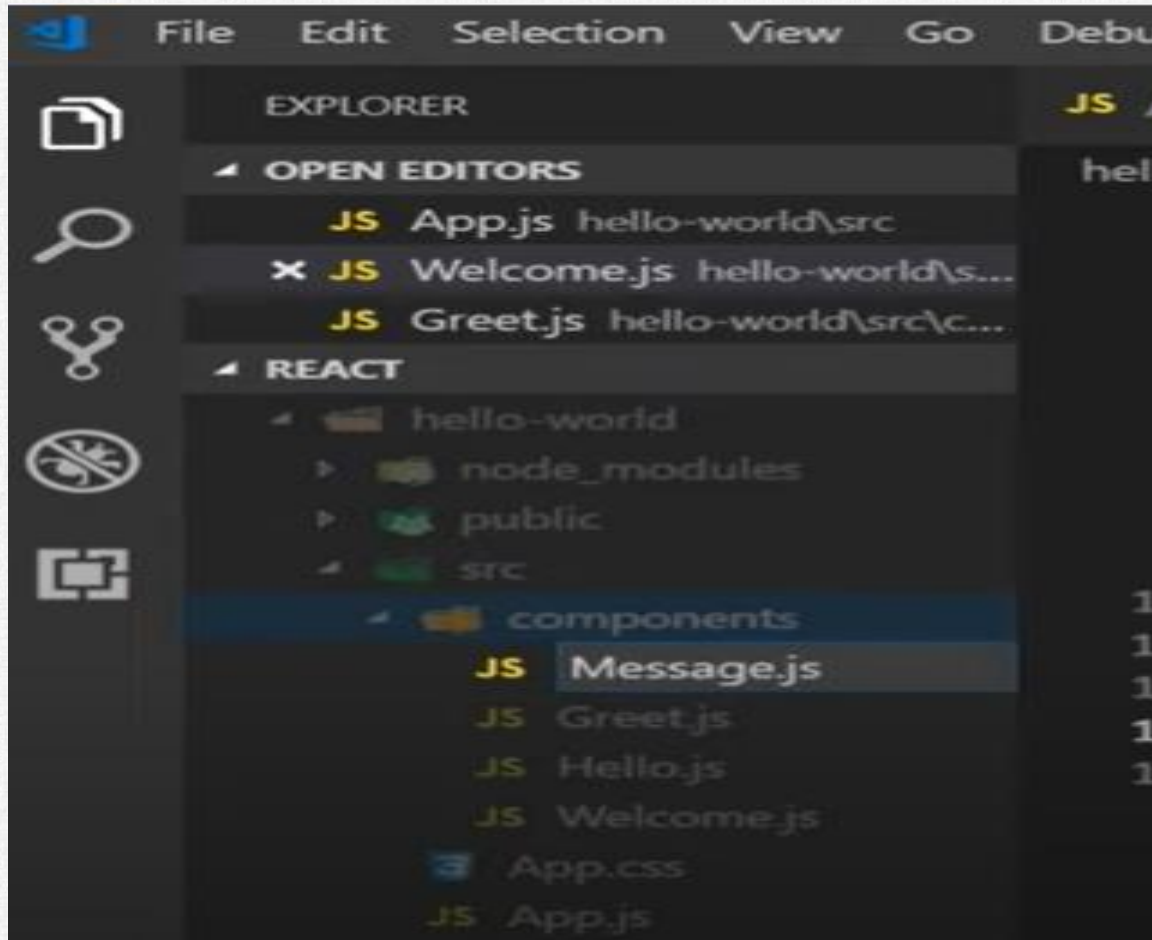
Update state on button click

I

- State is the internal object of the class and this is private ,not access outside of the class.

Create a new file inside the component folder

Message.js





JS App.js x JS Welcome.js JS Message.js x JS Greet.js



hello-world > src > components > JS Message.js > Message > render

```
1  import React, { Component } from 'react'
2
3  class Message extends Component {
4    render() {
5      return (
6        <h1>
7          Welcome visitor
8        </h1>
9      )
10   }
11 }
12
13 export default Message
14
```



JS App.js x JS Welcome.js JS Message.js JS Greet.js



hello-world > src > JS App.js > ...

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




File Edit Selection View Go Debug Terminal Help

Message.js - React - Visual Studio Code



JS App.js

JS Welcome.js

JS Message.js x

JS Greet.js



hello-world ▸ src ▸ components ▸ JS Message.js ▸ Message ▸ render

```
1  import React, { Component } from 'react'
2
3  class Message extends Component {
4    constructor() {
5      super()
6      this.state = {
7        message: 'Welcome visitor'
8      }
9    }
10
11    render() {
12      return <h1>{this.state.message}</h1>
13    }
14  }
15
16  export default Message
17
```





File Edit Selection View Go Debug Terminal Help

• Message.js - React - Visual Studio Code



JS App.js

JS Welcome.js

JS Message.js •

JS Greet.js



hello-world ▸ src ▸ components ▸ JS Message.js ▸ Message ▸ render

```
4   constructor() {
5     super()
6     this.state = {
7       message: 'Welcome visitor'
8     }
9   }
10
11  render() {
12    return (
13      <div>
14        <h1>{this.state.message}</h1>
15        <button>Subscribe</button>
16      </div>
17    )
18  }
19 }
20
21 export default Message
```

JS App.js

JS Welcome.js

JS Message.js x

JS Greet.js

hello-world ▸ src ▸ components ▸ JS Message.js ▸ Message ▸ render

```
7      message: 'Welcome Visitor'
8    }
9  }
10
11  changeMessage() {
12    this.setState({
13      message: 'Thank you for subscribing'
14    })
15  }
16
17  render() {
18    return (
19      <div>
20        <h1>{this.state.message}</h1>
21        <button onClick={() => this.changeMessage()}>Subscribe</button>
22      </div>
23    )
24  }
```

Welcome visitor

[Subscribe](#)

Thank you for subscribing

[Subscribe](#)

Ex:-state,setstate

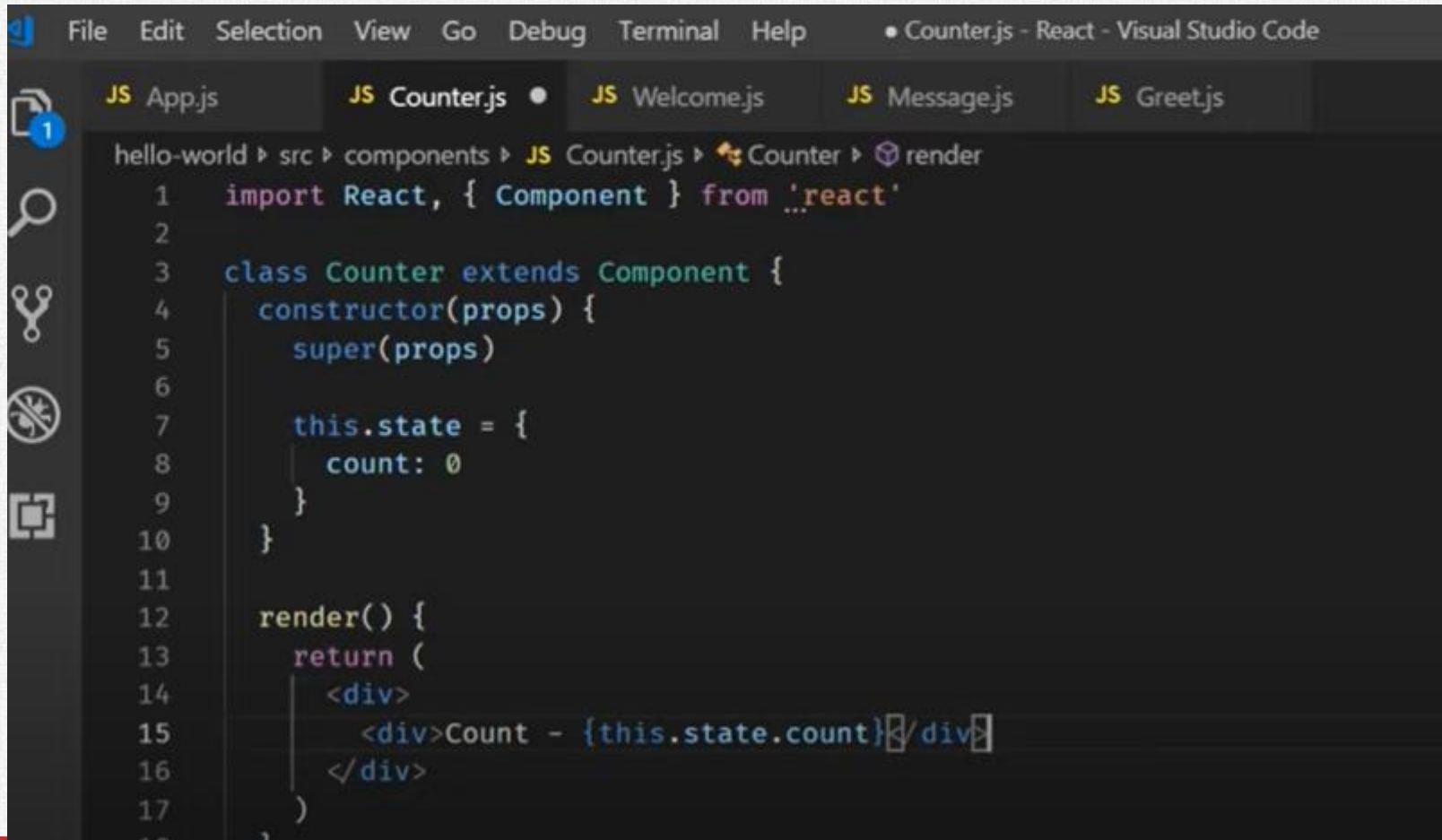
```
export default class Profile extends React.Component {  
  constructor() {  
    super();  
    this.state = {  
      name: 'peter',  
      email: 'peter@test.com',  
      count: 0  
    }  
  }  
    
  updateState() {  
    this.setState({  
      name: 'Bruce',  
      count: this.state.count+1  
    })  
  }  
}
```

Render call every time.

```
console.warn("render")
return (
  <div>
    <h1>Count {this.state.count}</h1>
    <h1>Email: {this.state.email}</h1>
    <button onClick={()=>{this.updateState()}} >Update
  </div>
)
}
```

Create a file Counter.js:-type rec and tab

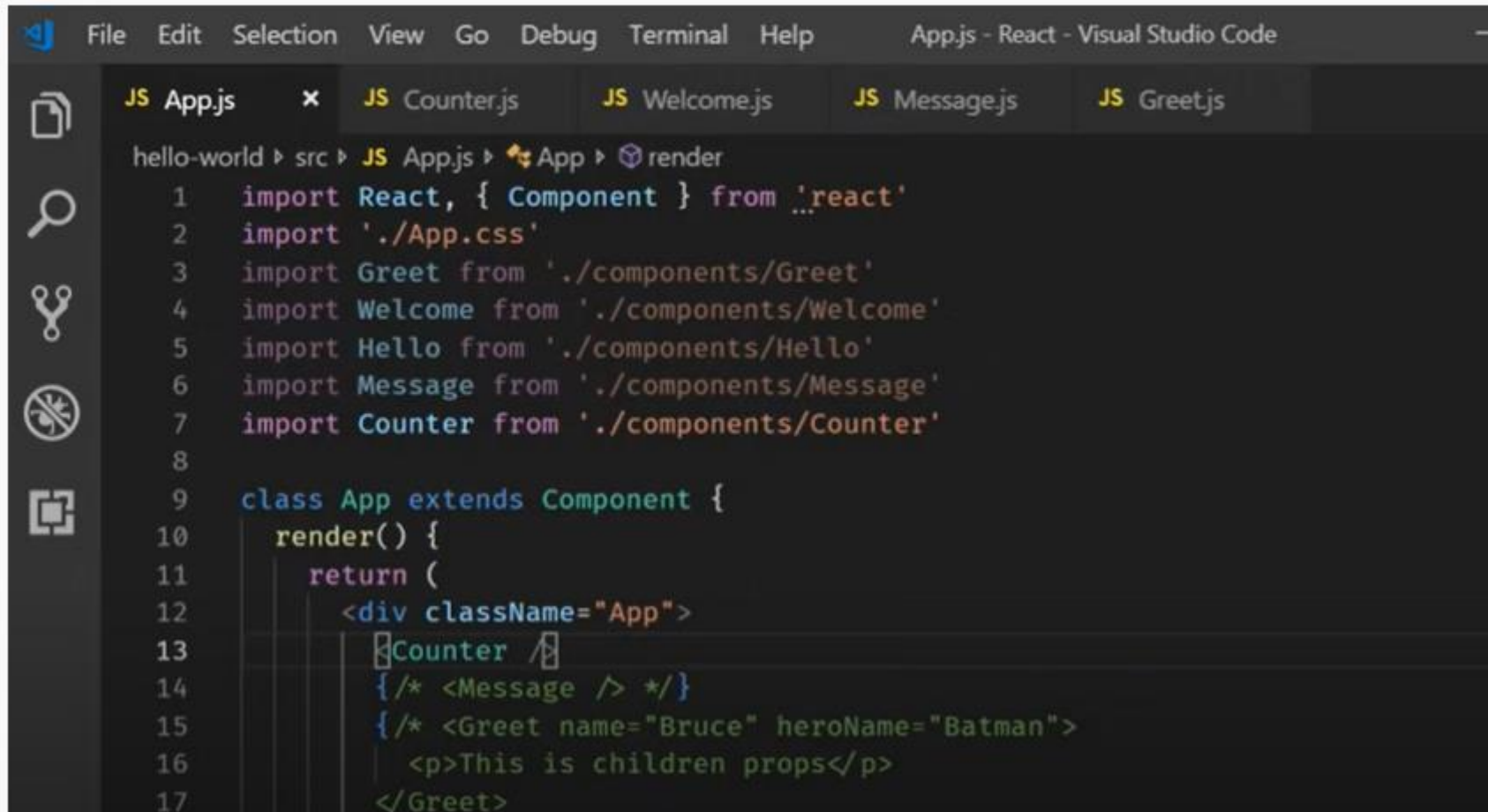
- Create code is created by default.
- Rconst for create a constructor

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar shows 'Counter.js - React - Visual Studio Code'. The Explorer sidebar on the left shows a file tree with 'App.js', 'Counter.js', 'Welcome.js', 'Message.js', and 'Greet.js'. The main editor area displays the code for 'Counter.js'. The code starts with an import statement for React and Component. It then defines a class 'Counter' that extends 'Component'. The class has a constructor that calls 'super(props)' and initializes 'this.state' with 'count: 0'. The 'render()' method returns a JSX element: a div containing 'Count - {this.state.count}'.

```
increment() {  
  this.state.count = this.state.count + 1  
  console.log(this.state.count)  
}  
  
render() {  
  return (  
    <div>  
      <div>Count - {this.state.count}</div>  
      <button onClick={() => this.increment()}>Increment</button>  
    </div>  
  )  
}
```

```
11  
12 increment() {  
13   this.setState({  
14     count: this.state.count + 1  
15   })  
16   console.log(this.state.count)  
17 }  
18
```


Calling Counter.js inside App.js

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar says 'App.js - React - Visual Studio Code'. The Explorer sidebar on the left shows a file tree with 'hello-world' > 'src' > 'App.js'. The Editor pane shows the code for 'App.js'. The code imports React, Component, and various components from './components/'. It defines a class App extending Component with a render method that returns a JSX element containing a Counter component and other UI elements.

```
File Edit Selection View Go Debug Terminal Help App.js - React - Visual Studio Code

JS App.js x JS Counter.js JS Welcome.js JS Message.js JS Greet.js

hello-world > src > JS App.js > App > render
1 import React, { Component } from 'react'
2 import './App.css'
3 import Greet from './components/Greet'
4 import Welcome from './components/Welcome'
5 import Hello from './components/Hello'
6 import Message from './components/Message'
7 import Counter from './components/Counter'
8
9 class App extends Component {
10   render() {
11     return (
12       <div className="App">
13         <Counter />
14         {/* <Message /> */}
15         {/* <Greet name="Bruce" heroName="Batman">
16           <p>This is children props</p>
17         </Greet>
```



File Edit Selection View Go Debug Terminal Help

Counter.js - React - Visual Studio Code



JS App.js

JS Counter.js x

JS Welcome.js

JS Message.js

JS Greet.js



hello-world > src > components > JS Counter.js > Counter > increment

```
11
12   increment() {
13     // this.setState(
14     //   {
15     //     count: this.state.count + 1
16     //   },
17     //   () => {
18     //     console.log('Callback value', this.state.count)
19     //   }
20     // )
21
22     this.setState(prevState => ({
23       count: prevState.count + 1
24     }))
25     console.log(this.state.count)
26   }
27
28   incrementFive() {
29     this.increment()
```

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.

Thanks