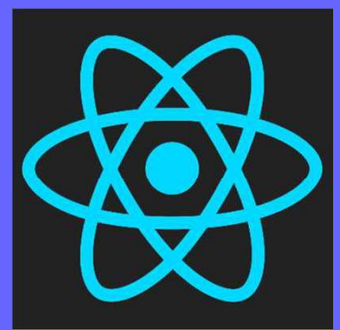
12

# Ref's HOC and Context

# Objectives

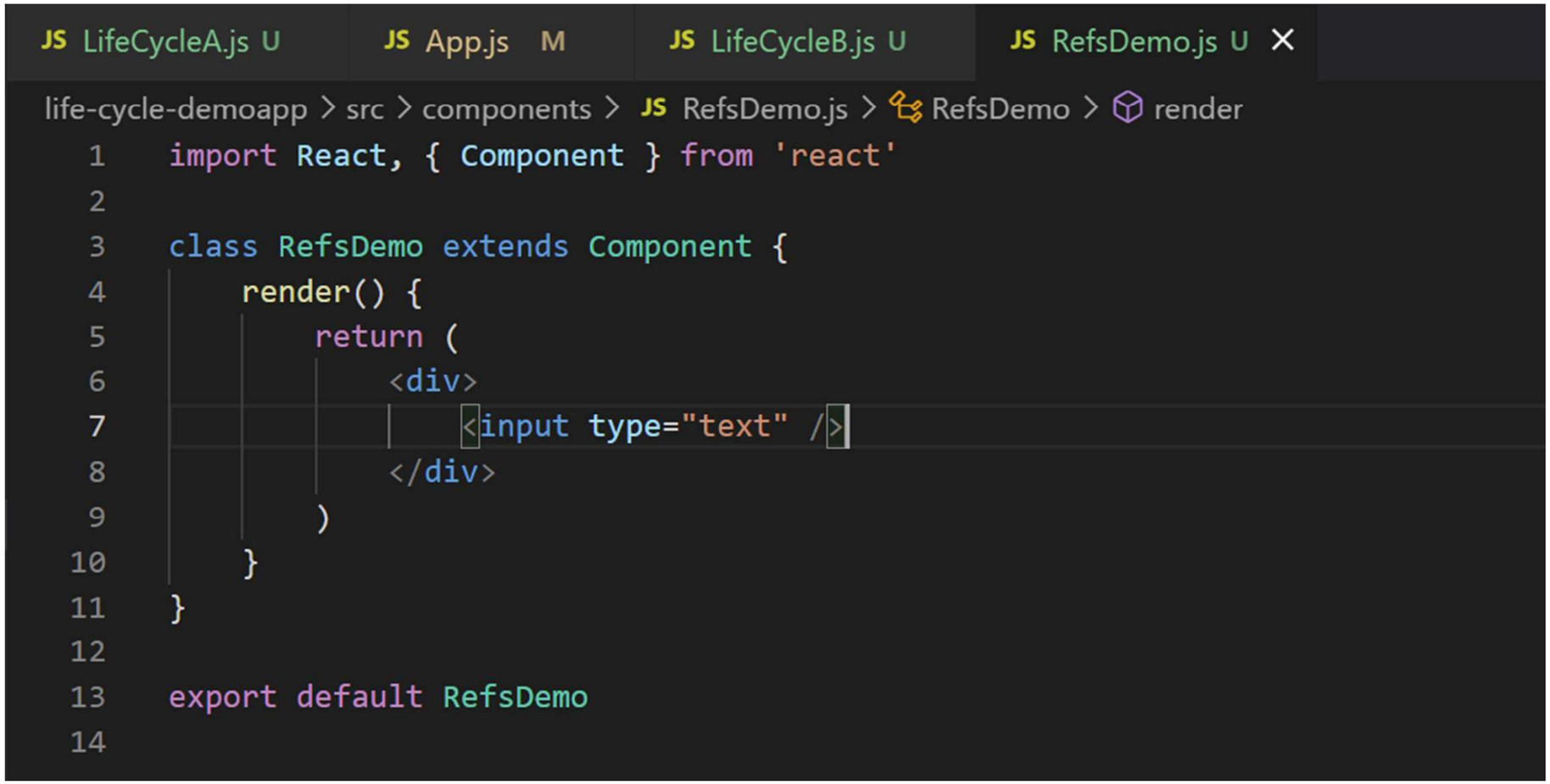
After completing this lesson, you should be able to do the following:

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



# Refs

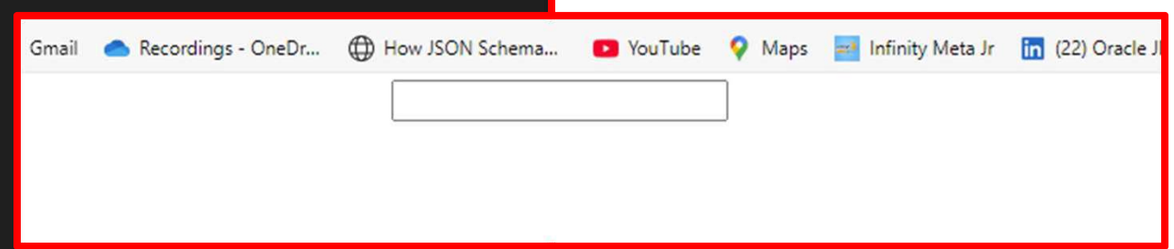
- Refs make it possible to access Dom nodes directly within react
- The more common use cases which is focusing a text input
- Suppose we have a login form as soon as the page is loaded let's say by default we want the user name input field to be focused


JS LifeCycleA.js U
JS App.js M
JS LifeCycleB.js U
JS RefsDemo.js U
life-cycle-demoapp > src > components > JS RefsDemo.js > RefsDemo > render
1 import React, { Component } from 'react'
2
3 class RefsDemo extends Component {
4 render() {
5 return (
6 
7 <input type="text" />
8 
9 )
10 }
11 }
12
13 export default RefsDemo
14

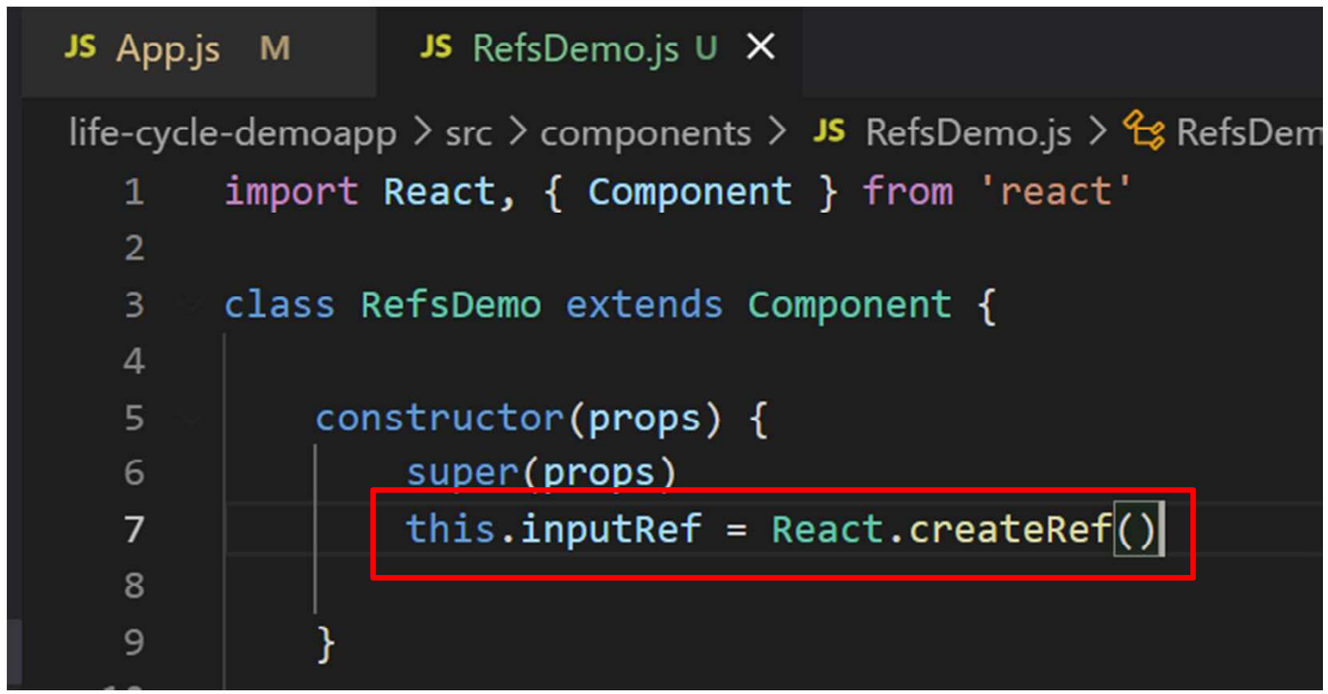
JS LifeCycleA.js U | JS App.js M | JS LifeCycleB.js U | JS RefsDemo.js U

life-cycle-demoapp > src > JS App.js > ...

```
1   import logo from './logo.svg';
2   import './App.css';
3   import LifeCycleA from './components/LifeCycleA';
4   import RefsDemo from './components/RefsDemo';
5   function App() {
6     return (
7       <div className="App">
8
9         <RefsDemo />
10        {/* <LifeCycleA /> */}
11      </div>
12    );
13  }
14
15  export default App;
```



- Our Requirement is as soon page is loaded we need Focus in the Text Field
- Using Refs we can do in 3 Simple Steps
  1. create a ref we do that using React.createRef()

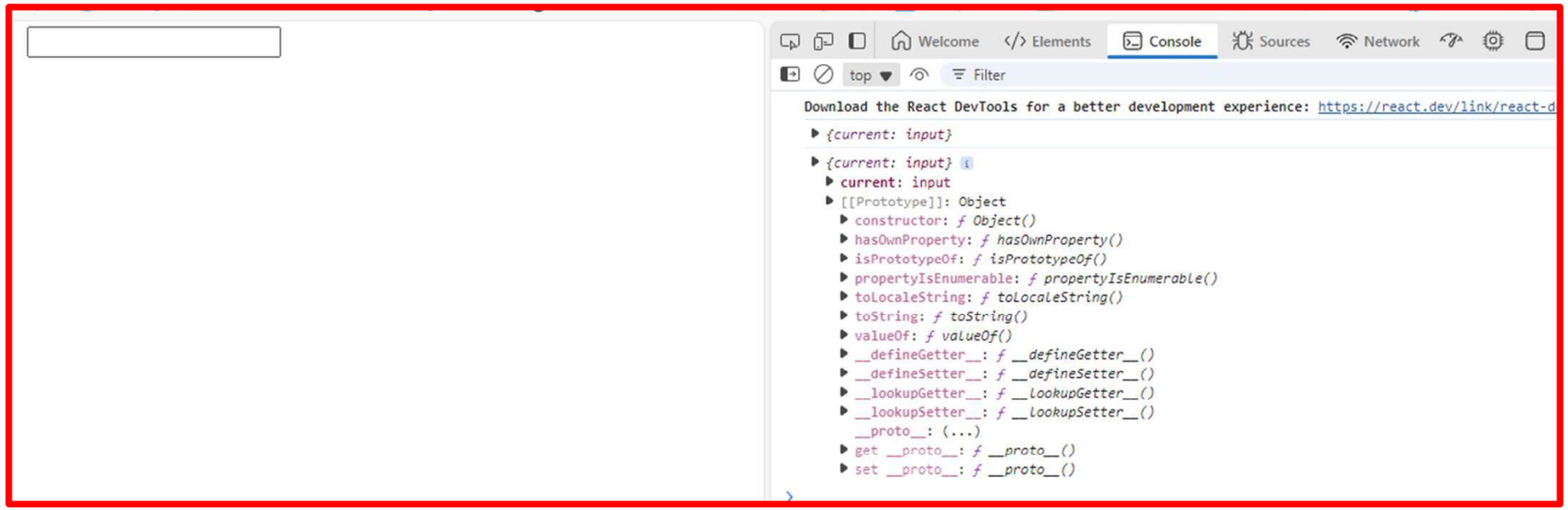


2. second step is to attach this ref to our input element in the render method and to attach a ref we make use of the reserved ref attribute

```
10
11      render() {
12          return (
13              <div>
14                  <input type="text" ref={this.inputRef}/>
15              </div>
16          )
17      }
18  }
```

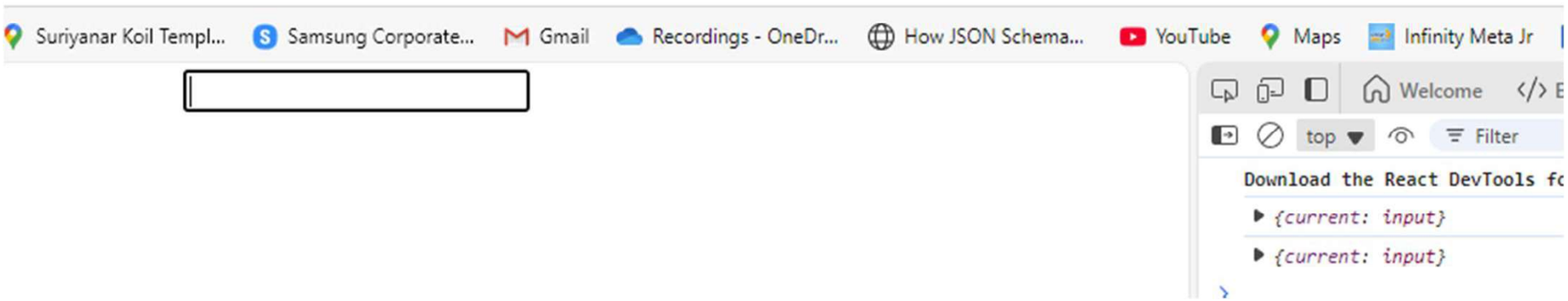
3. The third and final step is to call the focus method on this input element

```
componentDidMount() {
    console.log(this.inputRef)
}
```



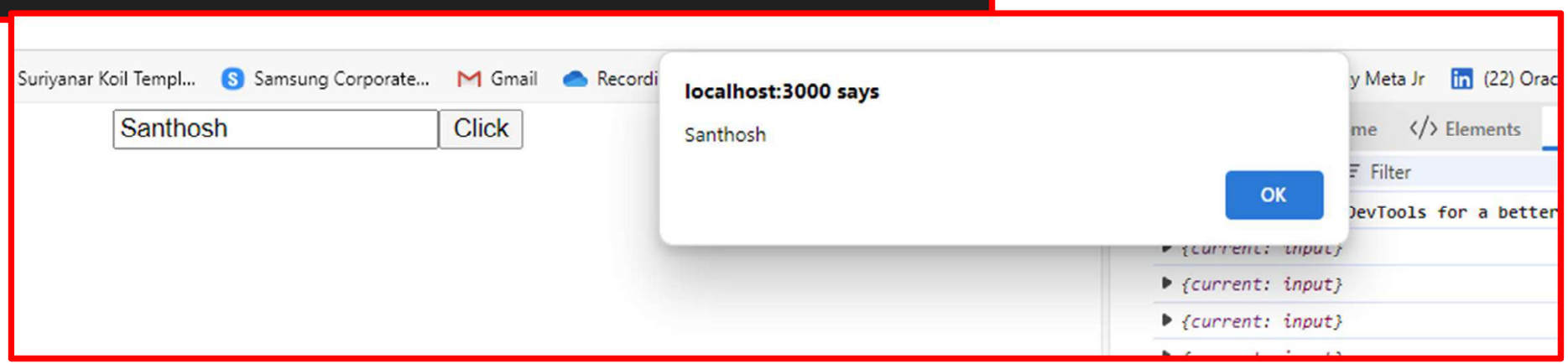
# Now on Page Load the Component has Got the Focus

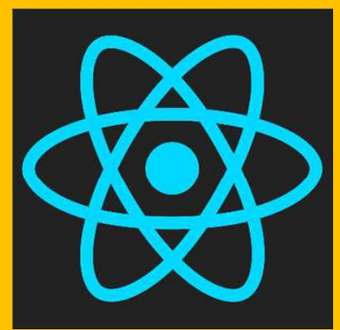
```
8       }
9
10      componentDidMount() {
11          this.inputRef.current.focus()
12          console.log(this.inputRef)
13      }
14
```



# Second Usecase of Using Refs is to Fetch the Value

```
15      clickHandler = () => {
16          alert(this.inputRef.current.value)
17      }
18
19      render() {
20          return (
21              <div>
22                  <input type="text" ref={this.inputRef}/>
23                  <button onClick={this.clickHandler}>Click</button>
24              </div>
25          )
26      }
27  }
```



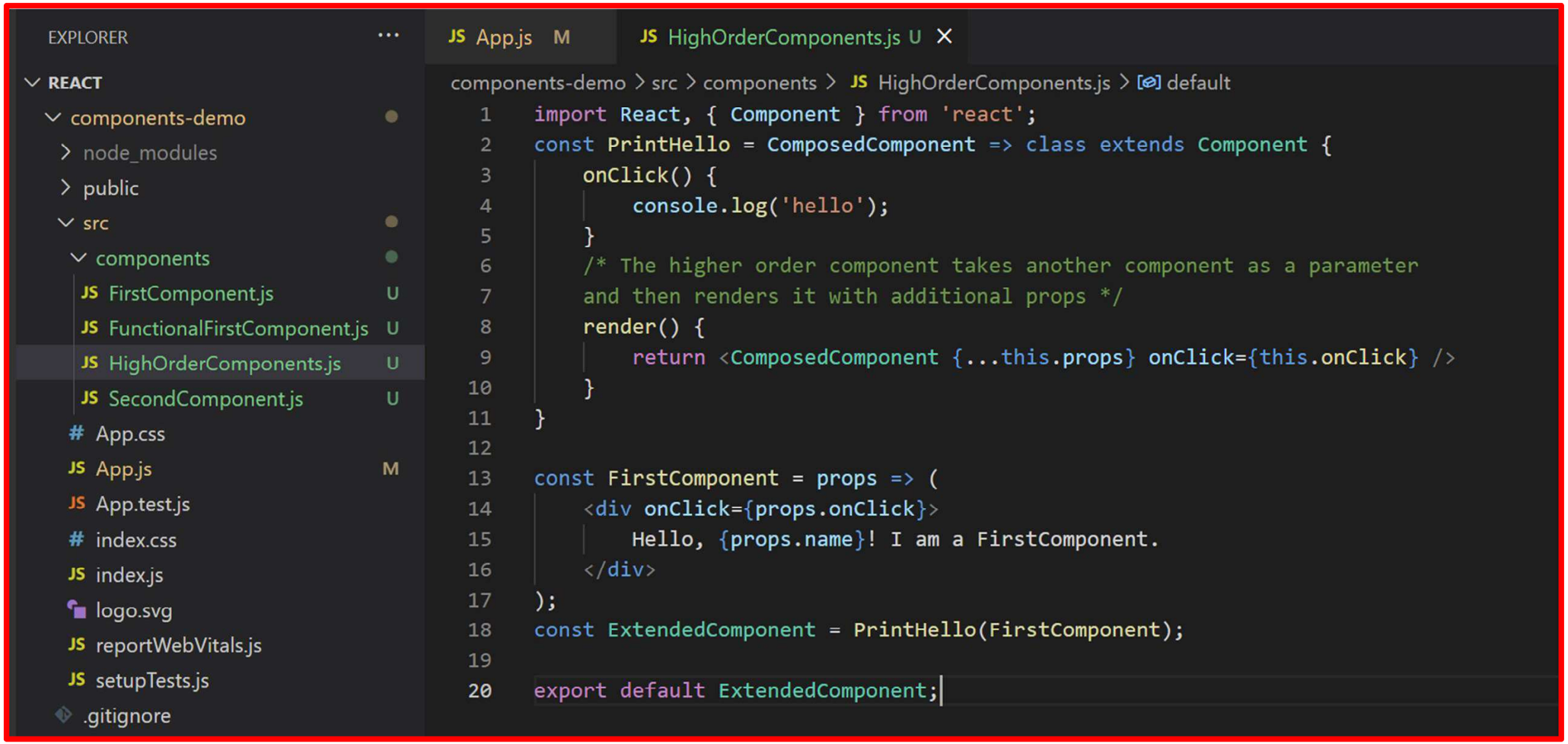


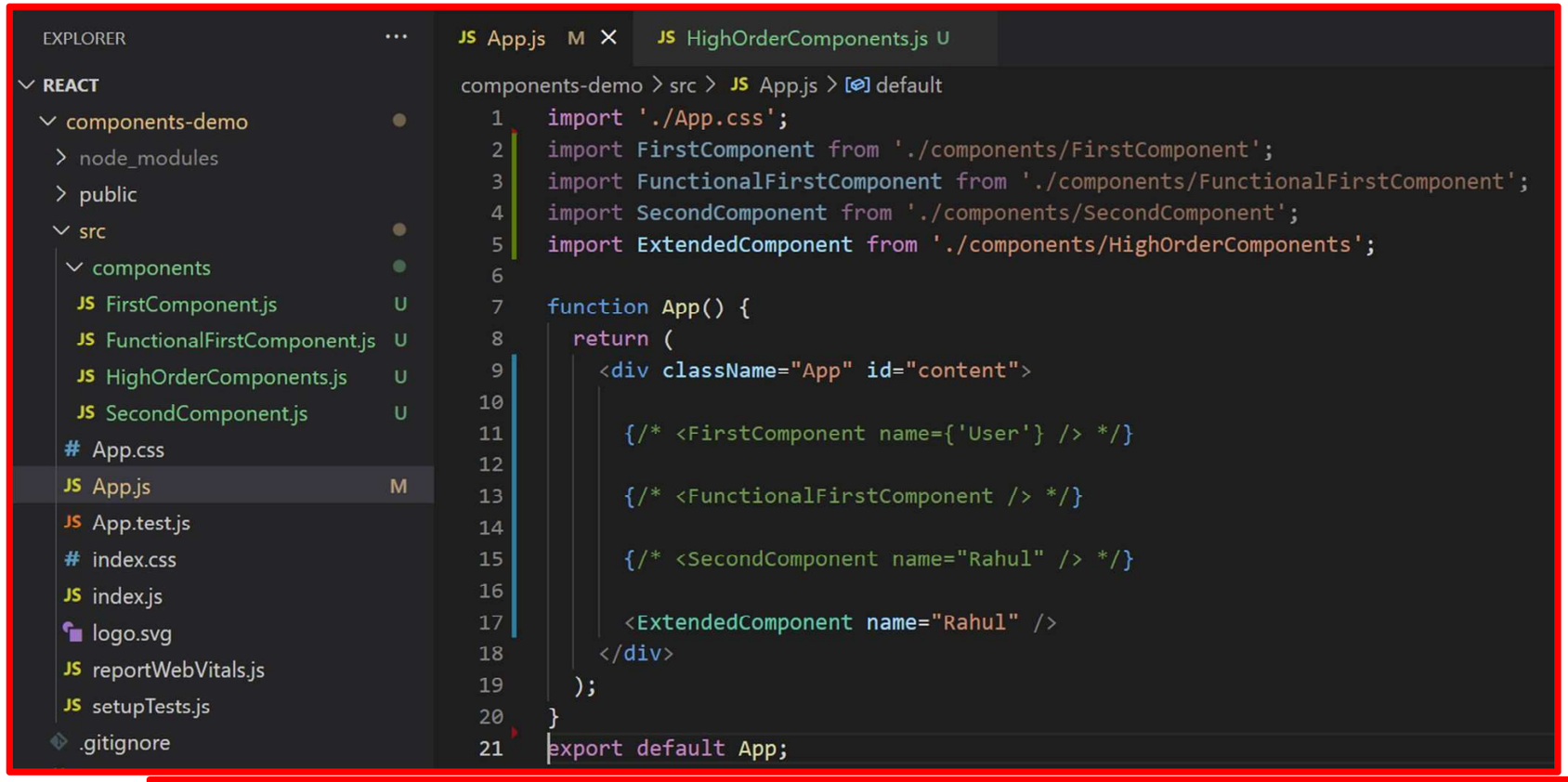
# HOC

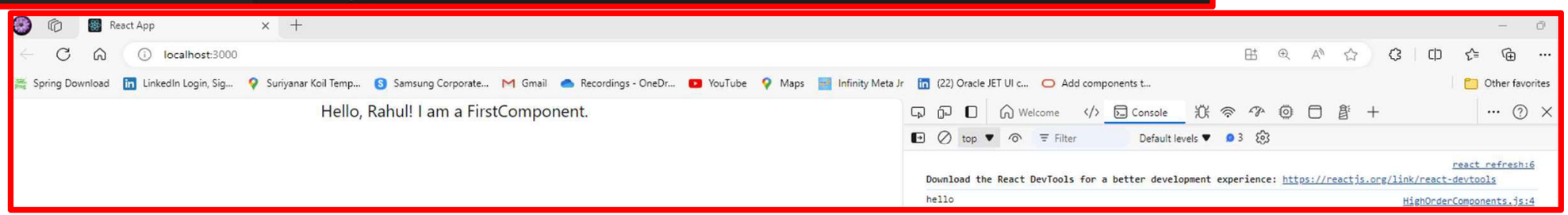
# Higher Order Components

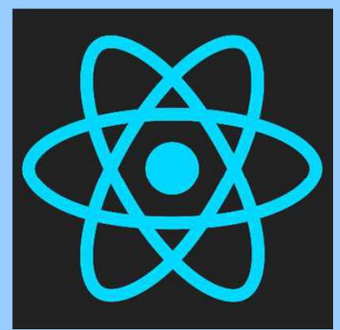
- Higher order components (HOC) allow to share component functionality.

- Higher order components are used when you want to share logic across several components regardless of how different they render.

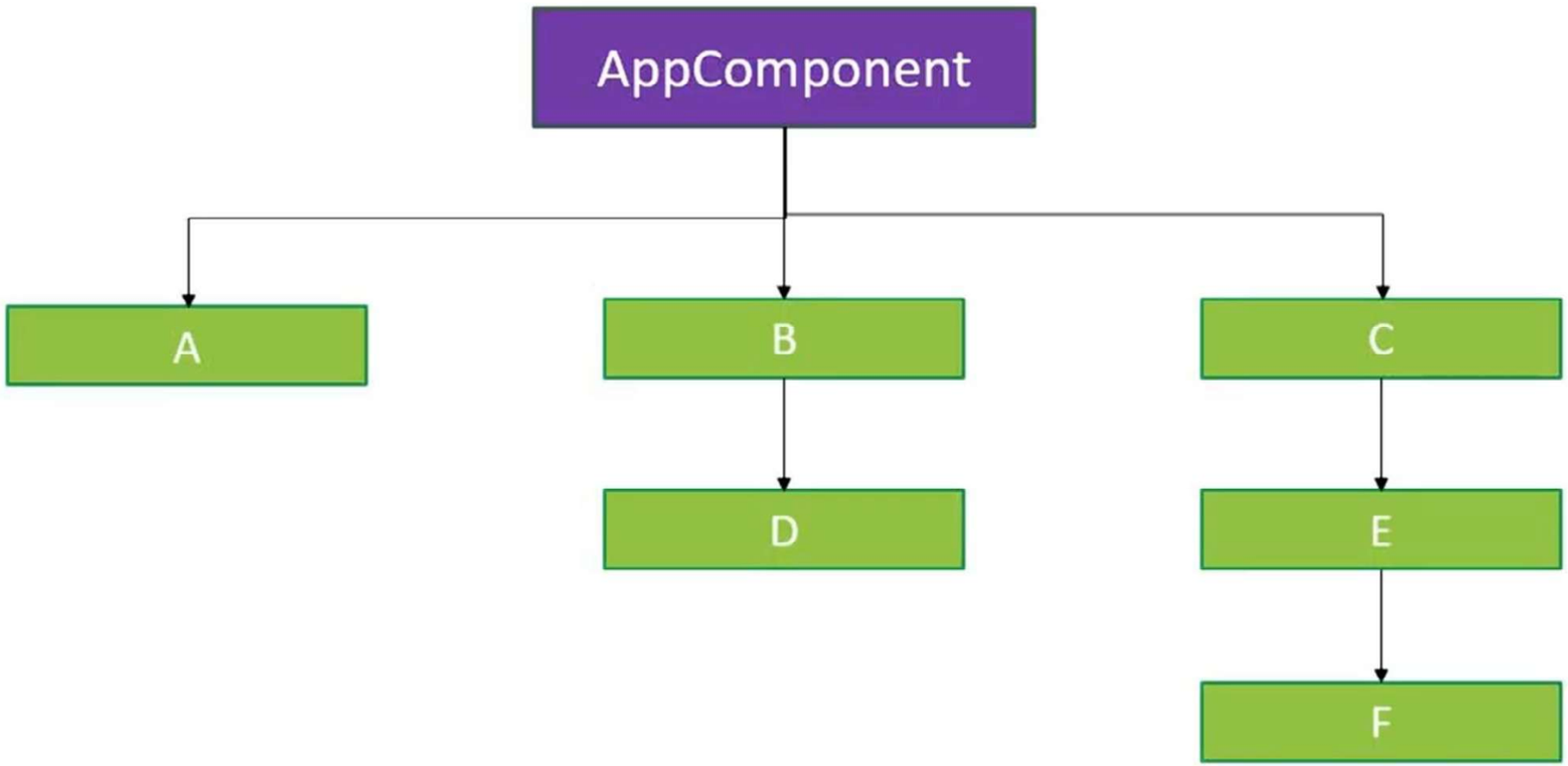

EXPLORER
REACT
components-demo
node_modules
public
src
components
FirstComponent.js U
FunctionalFirstComponent.js U
HighOrderComponents.js U
SecondComponent.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
App.js M
HighOrderComponents.js U
components-demo > src > components > HighOrderComponents.js > default
1 import React, { Component } from 'react';
2 const PrintHello = ComposedComponent => class extends Component {
3 onClick() {
4 console.log('hello');
5 }
6 /* The higher order component takes another component as a parameter
7 and then renders it with additional props */
8 render() {
9 return <ComposedComponent {...this.props} onClick={this.onClick} />
10 }
11 }
12
13 const FirstComponent = props => (
14 
15 Hello, {props.name}! I am a FirstComponent.
16 
17 );
18 const ExtendedComponent = PrintHello(FirstComponent);
19
20 export default ExtendedComponent;


EXPLORER
REACT
components-demo
node_modules
public
src
components
FirstComponent.js U
FunctionalFirstComponent.js U
HighOrderComponents.js U
SecondComponent.js U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
App.js M
HighOrderComponents.js U
components-demo > src > App.js > default
1 import './App.css';
2 import FirstComponent from './components/FirstComponent';
3 import FunctionalFirstComponent from './components/FunctionalFirstComponent';
4 import SecondComponent from './components/SecondComponent';
5 import ExtendedComponent from './components/HighOrderComponents';
6
7 function App() {
8 return (
9 
10
11 {/* <FirstComponent name={'User'} /> */}
12
13 {/* <FunctionalFirstComponent /> */}
14
15 {/* <SecondComponent name="Rahul" /> */}
16
17 <ExtendedComponent name="Rahul" />
18 
19 );
20 }
21 export default App;

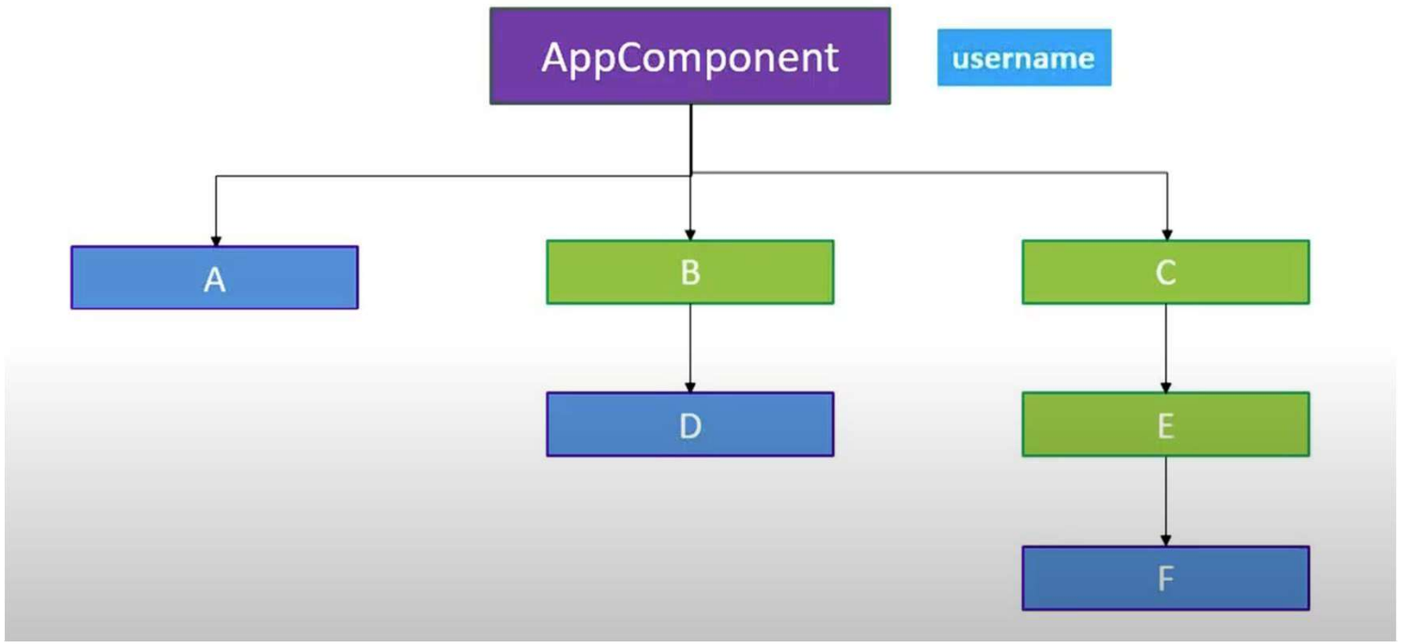

React App
localhost:3000
Hello, Rahul! I am a FirstComponent.
Welcome
Console
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
hello
HighOrderComponents.js:4



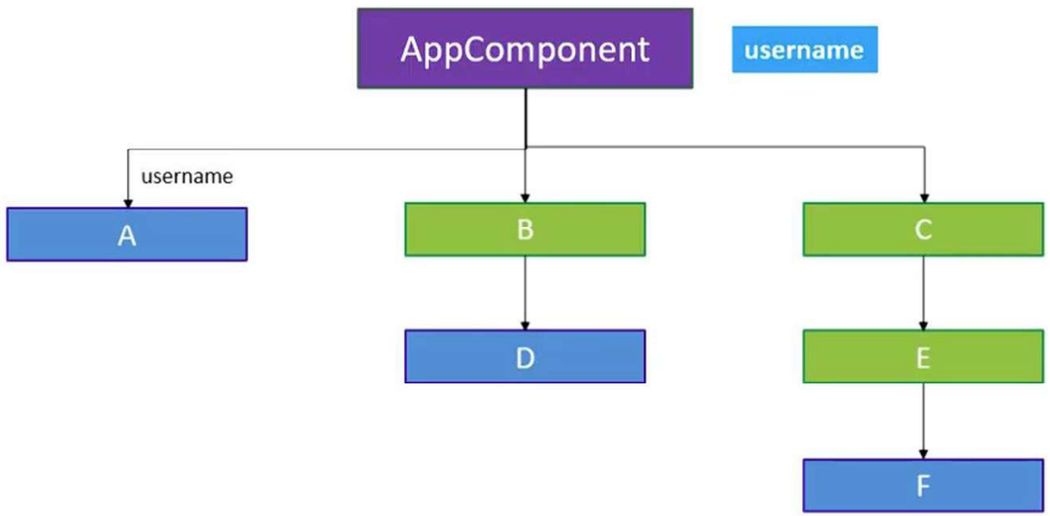
# Context


AppComponent
A
B
C
D
E
F

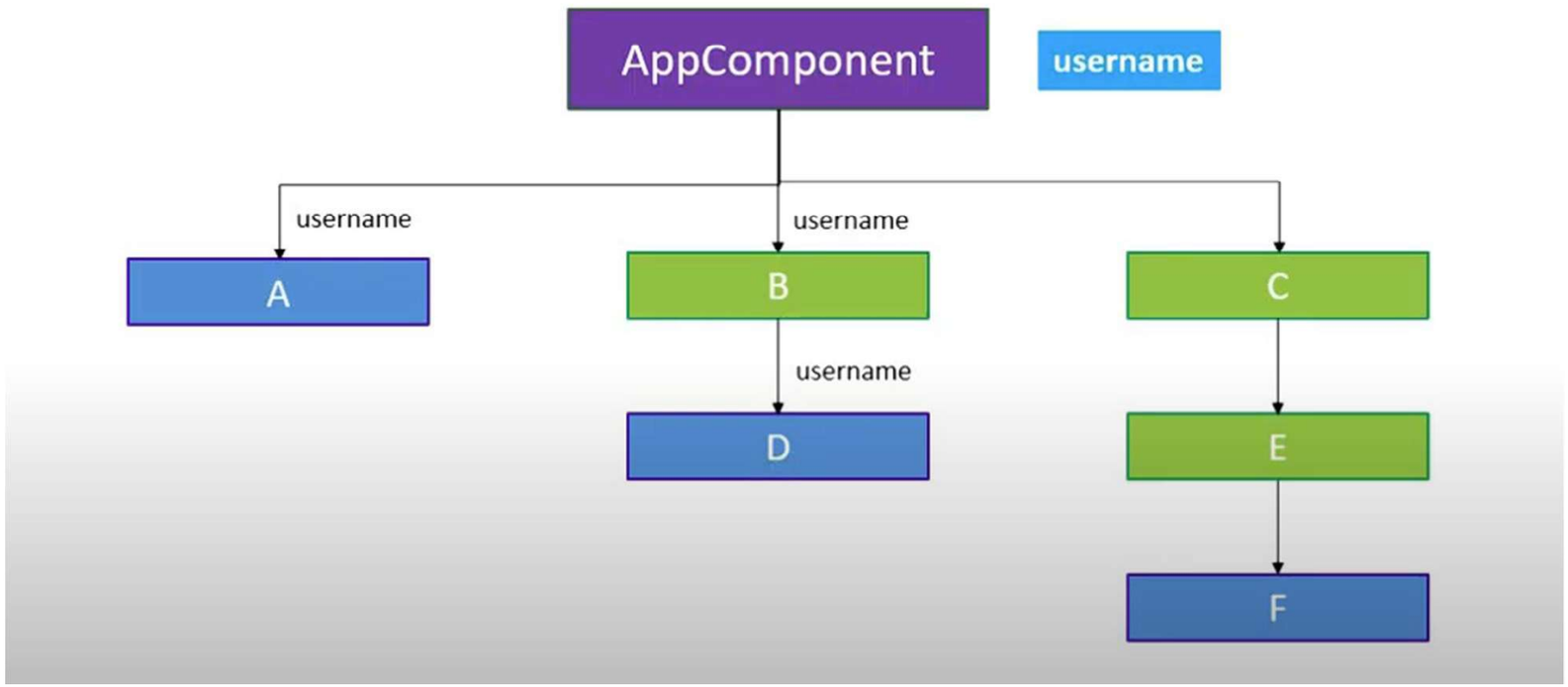
- The requirement in our application is that the components **A  D and F** are supposed to display the logged in user name that information is maintained as a property in the App component so to be able to display the user name in the nested components we need to pass down the username as a prop for component A



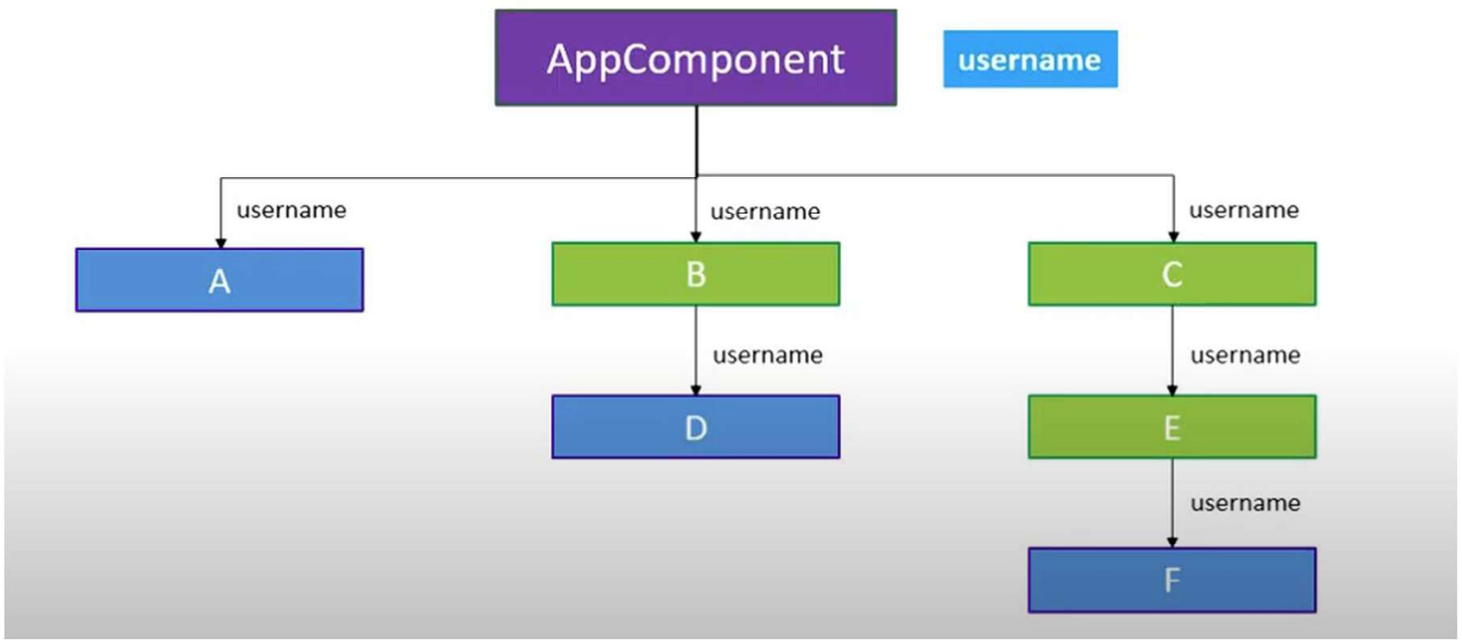
- For ComponentA is straightforward Pass it as props



- For component D however we have the intermediate component B so we have to pass down the user name as a prop to component B and that in turn has to pass down the prop to component D



- The scenario is somewhat similar for component F as well the prop has to be passed through component C and then component E and then finally to component F



- Even though components B  C and E do not need the prop we still have to send the prop through them to be able to pass it to components further down in the tree
- Imagine if the component were to be nested five or ten levels deep all the components in between would have to forward the prop this especially becomes a problem

- What would be nice is if we could directly send data to the required component without having to manually drill down the props through every level of the component tree
- This is where **CONTEXT** comes into picture
- **Context** provides a way to pass data through the component tree without having to pass props down manually at every level

# Summary

In this lesson, you should have learned how to:

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