REACTJS – MODULE 5

HIGHER ORDER COMPONENT(HOC)

accenture

AGENDA

- Understand what is higher order component
- How to achieve HOC
- Benefits of HOC
- Demo
- Activity

accenture

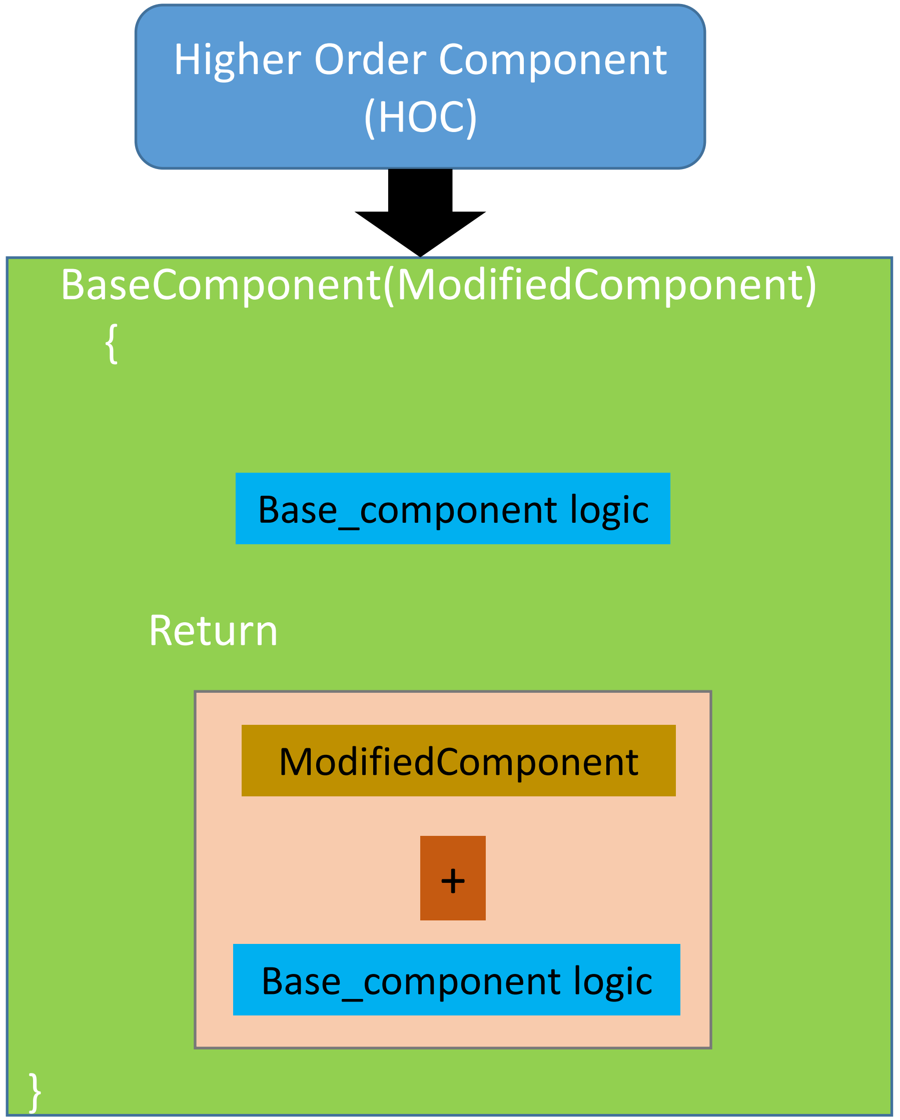
Introduction to Higher Order Component (HOC)

React allows us one of the technique for creating new component by reusing existing components logic

The component which perform this technique are known as higher order component(HOC)

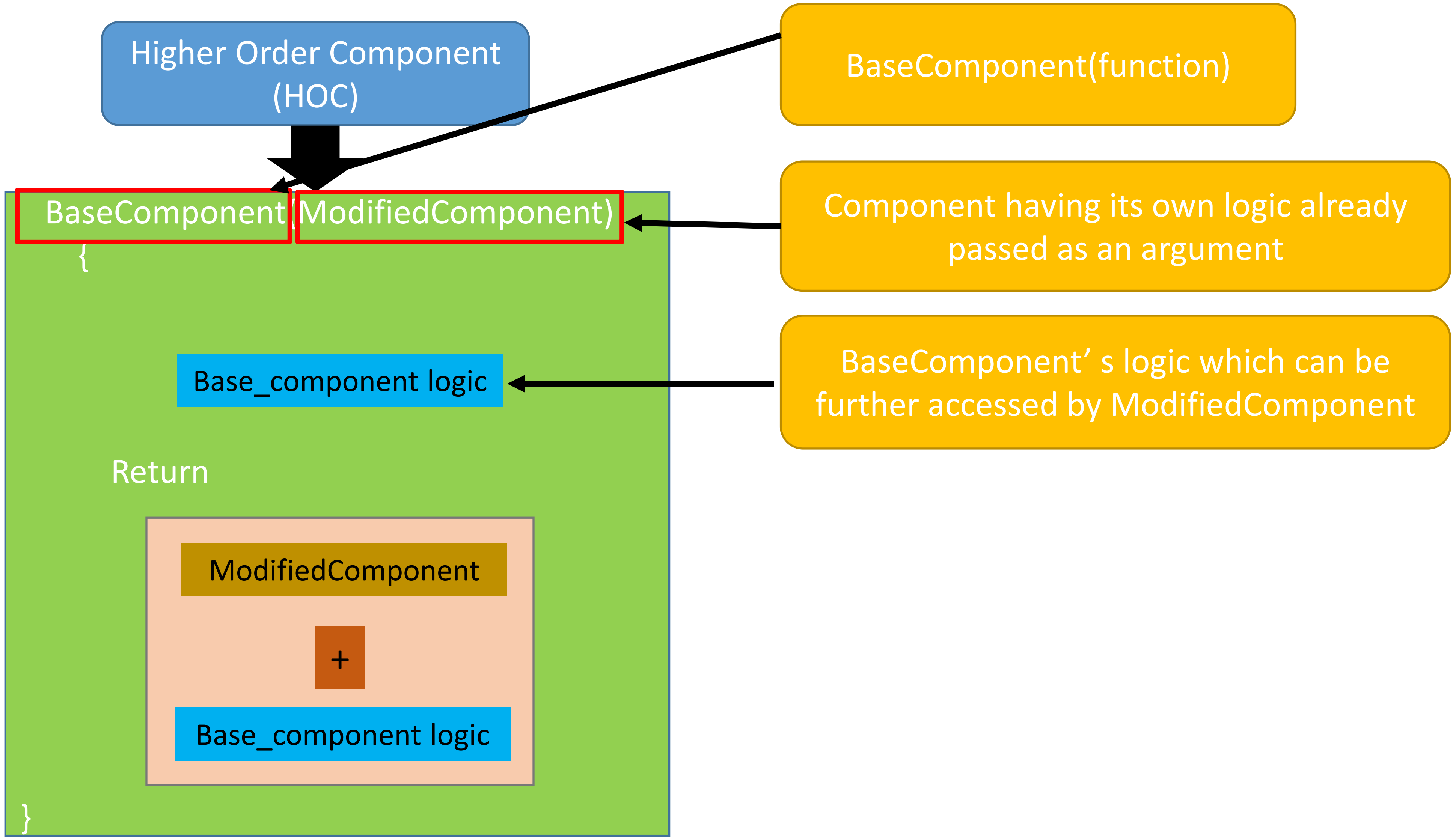
HOC are like a function which takes component as a input and return modified component as a output with additional features

Introduction to Higher Order Component (HOC)



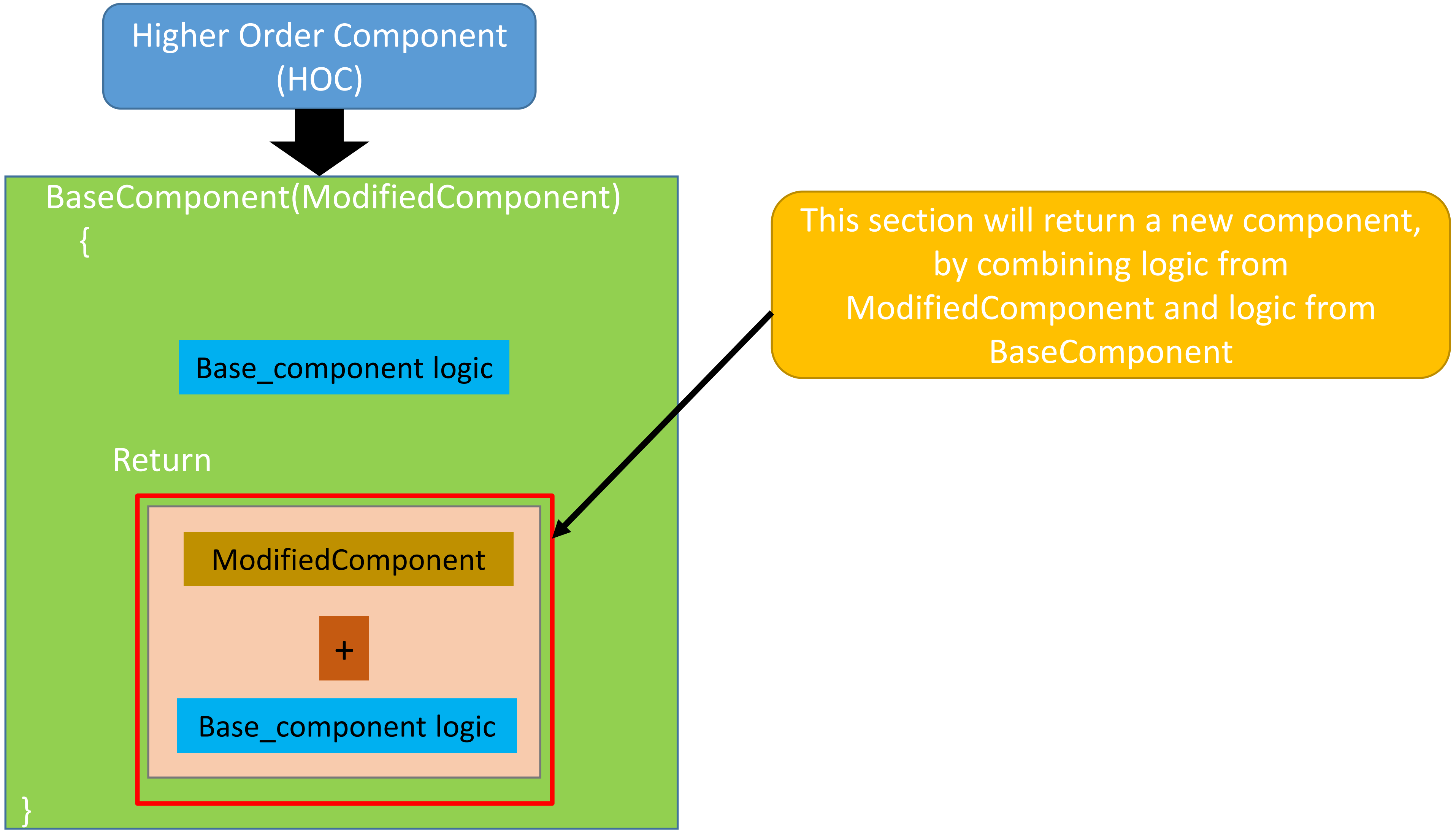
Here HOC is accepting a component as an argument and returns the modified component by adding/extending some logic

Introduction to Higher Order Component (HOC)

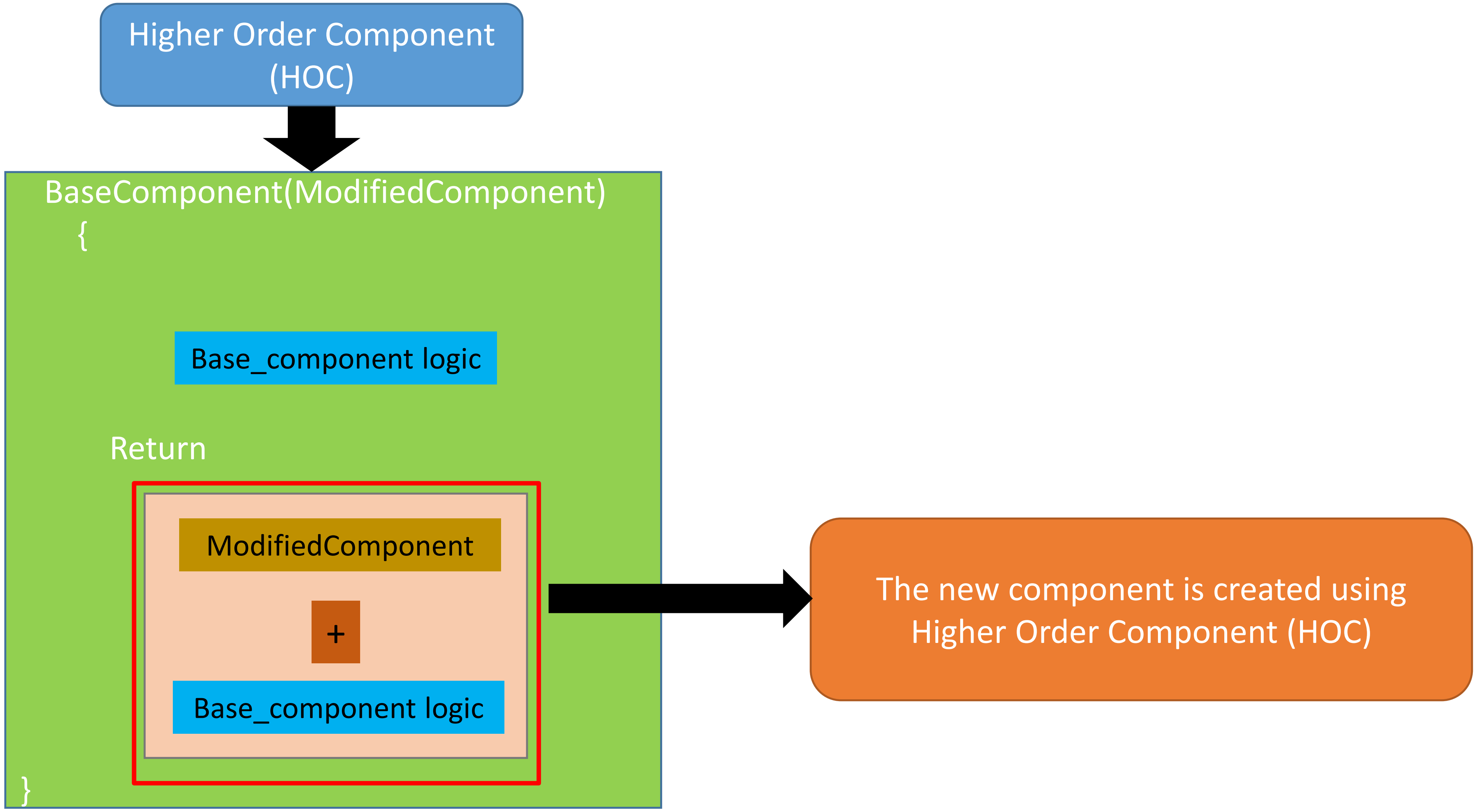


Higher Order Component (HOC)
BaseComponent(ModifiedComponent)
{
Base_component logic
Return
ModifiedComponent
+
Base_component logic
}
BaseComponent(function)
Component having its own logic already passed as an argument
BaseComponent' s logic which can be further accessed by ModifiedComponent

Introduction to Higher Order Component (HOC)


Higher Order Component (HOC)
BaseComponent(ModifiedComponent)
{
Base_component logic
Return
ModifiedComponent
+
Base_component logic
}
This section will return a new component, by combining logic from ModifiedComponent and logic from BaseComponent

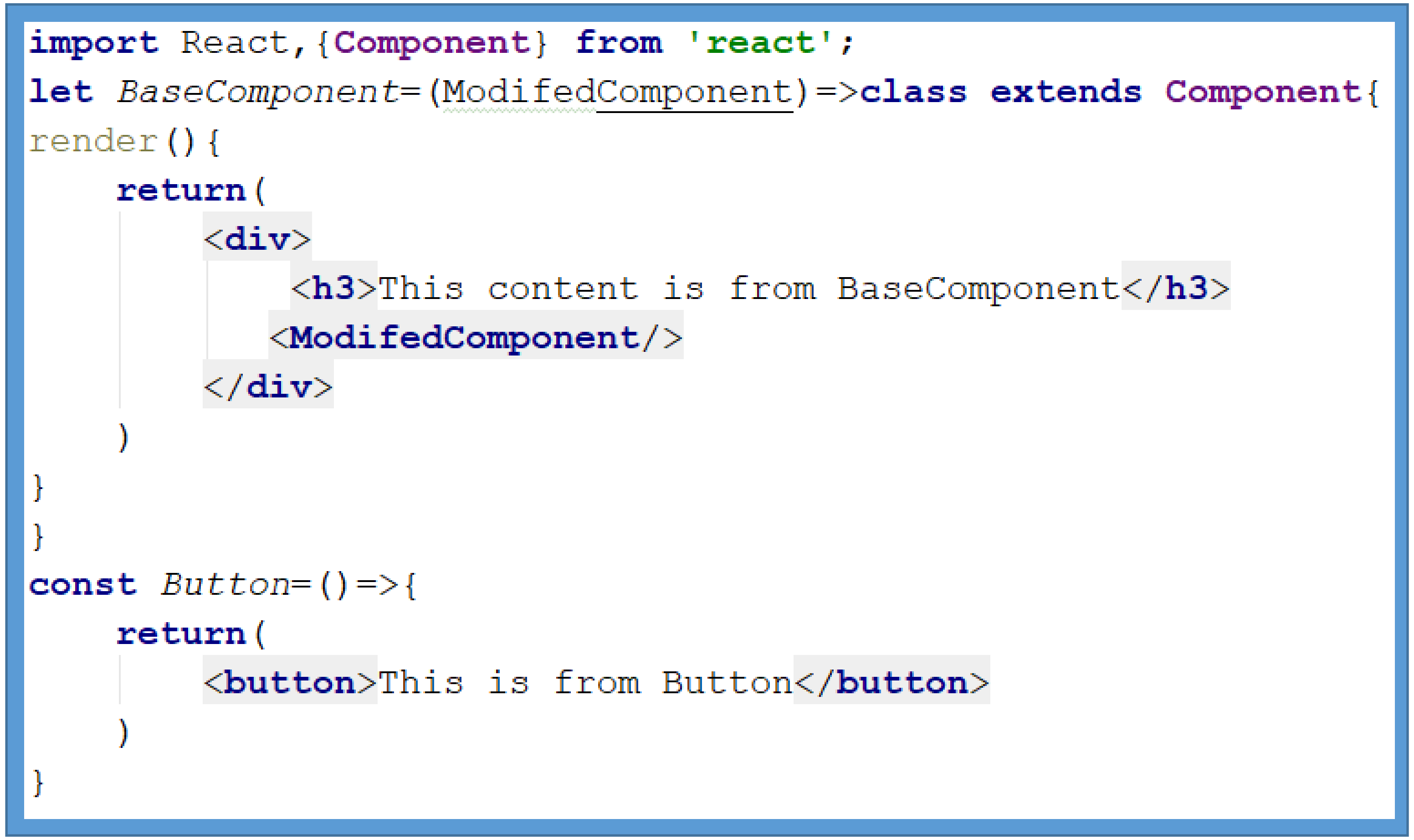
Introduction to Higher Order Component (HOC)



Higher Order Component (HOC)
BaseComponent(ModifiedComponent)
{
Base_component logic
Return
ModifiedComponent
+
Base_component logic
}
The new component is created using Higher Order Component (HOC)

Demo – Higher Order Component

Create a new class BaseComponent.js



```
import React,{Component} from 'react';
let BaseComponent=(ModifedComponent)=>class extends Component{
render(){
    return(
        <div>
            <h3>This content is from BaseComponent</h3>
           <ModifedComponent/>
        </div>
    )
}
}
const Button=()=>{
    return(
        <button>This is from Button</button>
    )
}
```

Demo – Higher Order Component (Cont …)

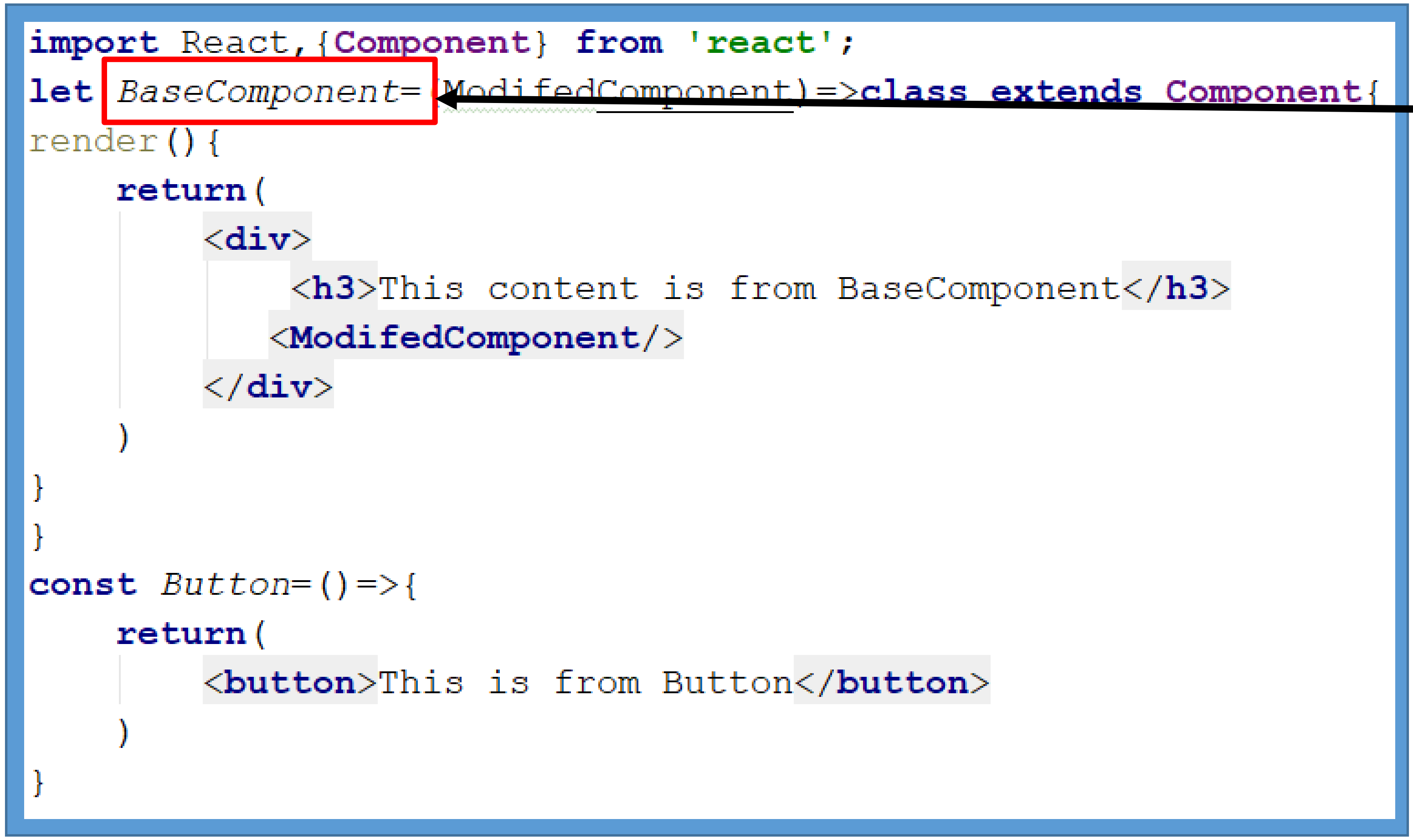
```
let NewButton=BaseComponent(Button);

const Label=()=>{
    return(
        <label>This is from Label</label>
    )
}
let NewLabel=BaseComponent(Label);
class Container extends Component{
    render(){
        return(
            <div>
                <NewButton/>
                <NewLabel/>
            </div>
        )
    }
}
```

```
export default Container;
```

Demo – Higher Order Component (Cont …)

Create a new class BaseComponent.js

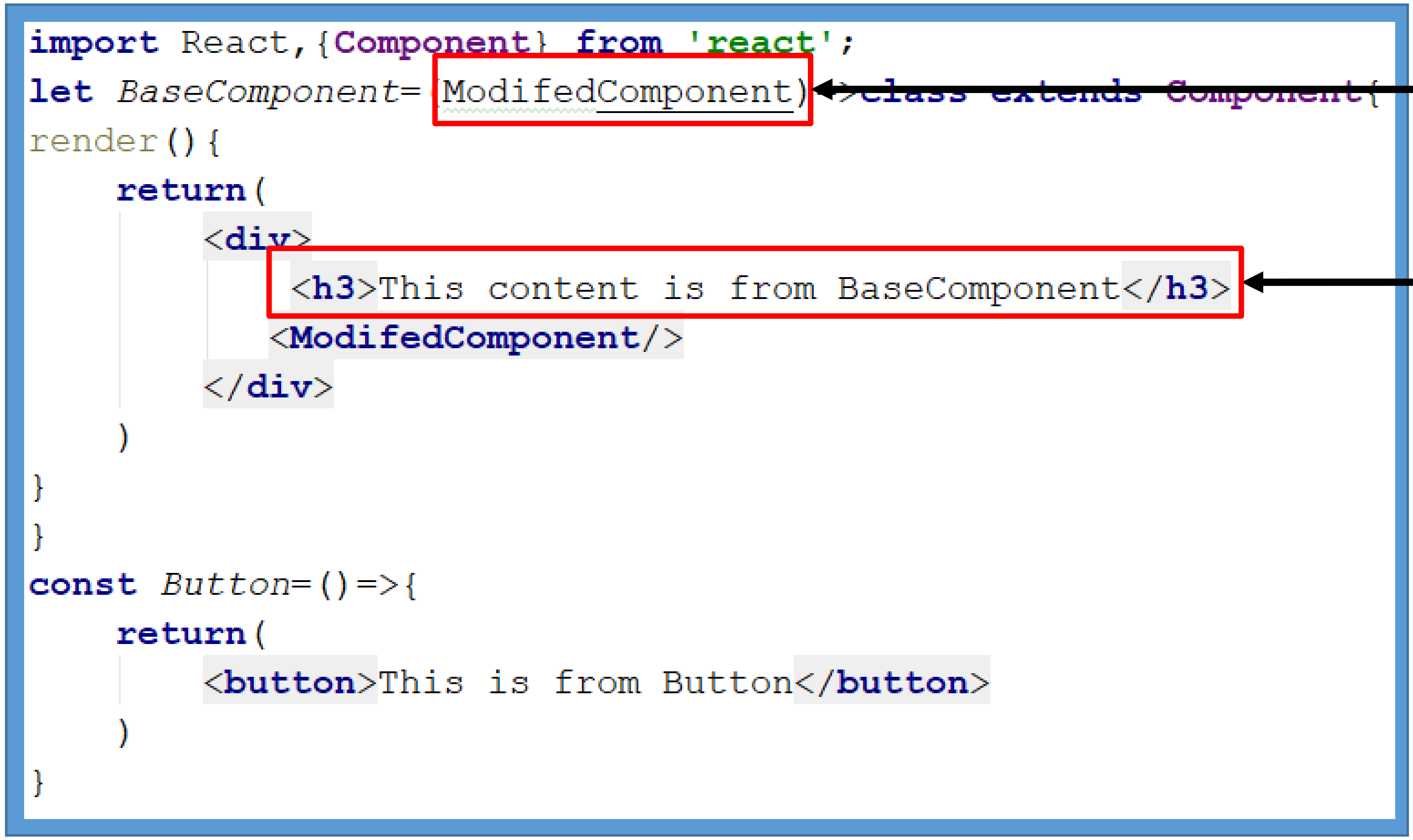


```
import React,{Component} from 'react';
let BaseComponent= (ModifedComponent)=>class extends Component{
render(){
    return(
        <div>
            <h3>This content is from BaseComponent</h3>
          <ModifedComponent/>
        </div>
    )
}
}
const Button=()=>{
    return(
        <button>This is from Button</button>
    )
}
```

BaseComponent(it is a function). This acts like a Higher order component

Demo – Higher Order Component (Cont …)

Create a new class BaseComponent.js



```
import React,{Component} from 'react';
let BaseComponent=(ModifedComponent)=>class extends Component{
render(){
    return(
        <div>
            <h3>This content is from BaseComponent</h3>
            <ModifedComponent/>
        </div>
    )
}
}
const Button=()=>{
    return(
        <button>This is from Button</button>
    )
}
```

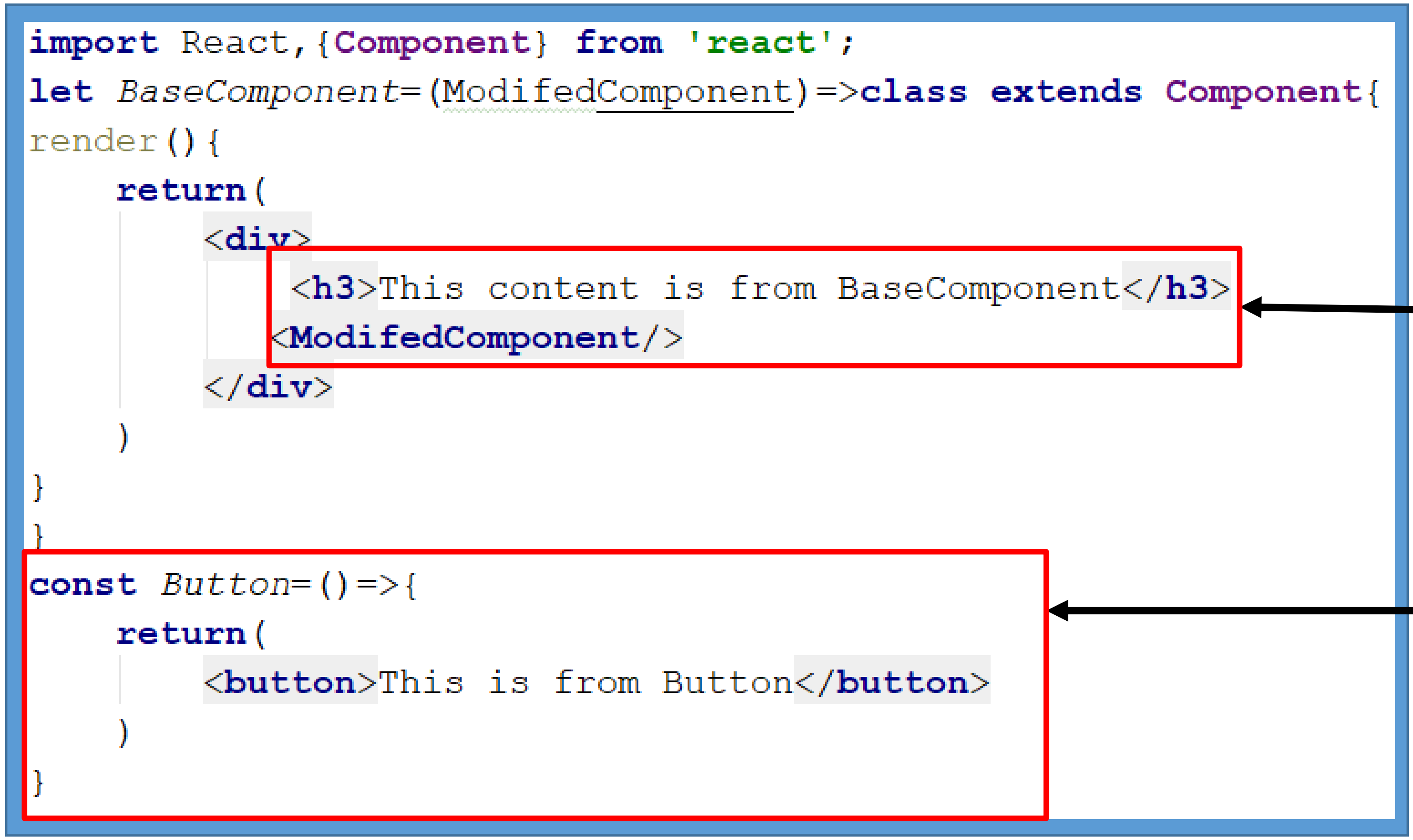
This is name of a parameter(component) passed to BaseComponent(HOC)

This is BaseComponent’ s logic

accenture

Demo – Higher Order Component (Cont …)

Create a new class BaseComponent.js



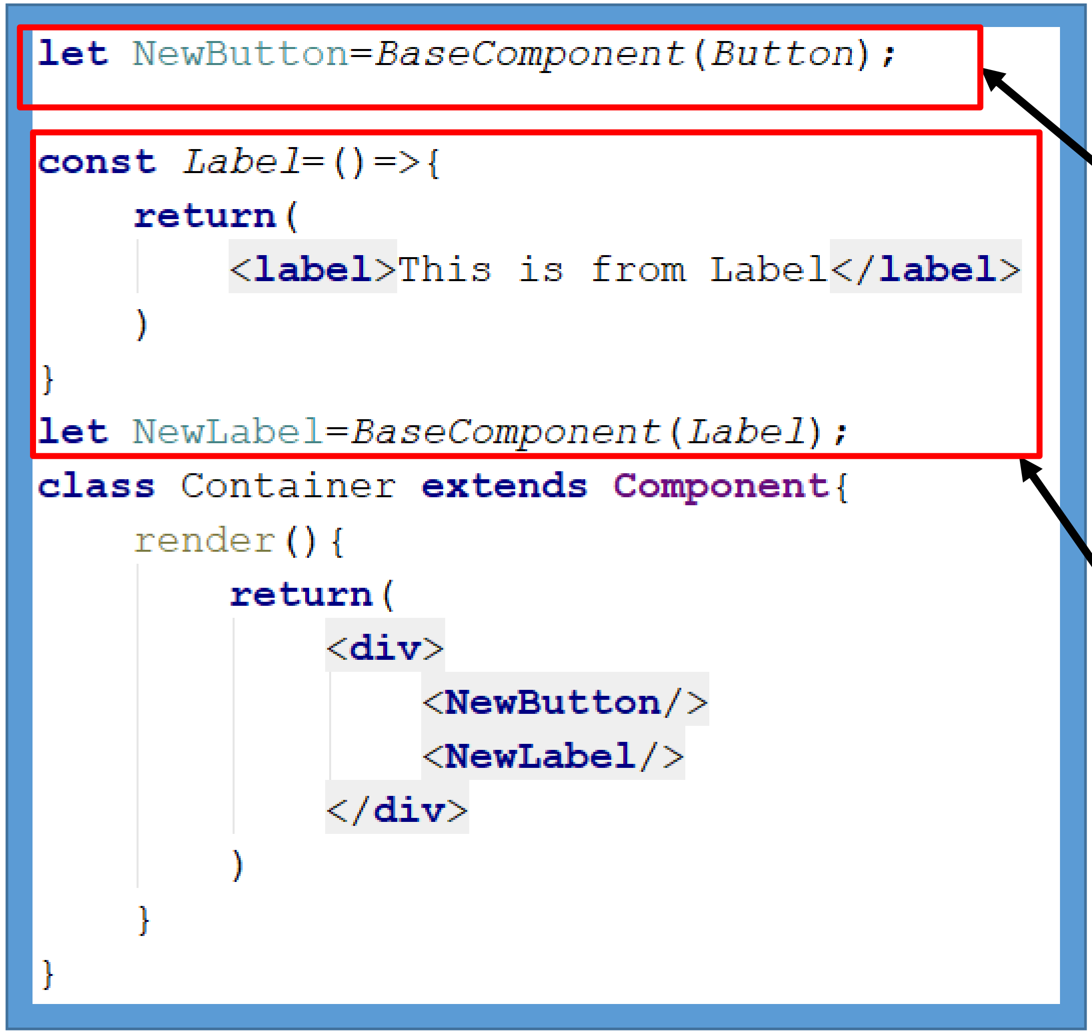
```
import React,{Component} from 'react';
let BaseComponent=(ModifedComponent)=>class extends Component{
render(){
    return(
        <div>
            <h3>This content is from BaseComponent</h3>
        <ModifedComponent/>
        </div>
    )
}
}
const Button=()=>{
    return(
        <button>This is from Button</button>
    )
}
```

BaseComponent' s logic is combined with other component which was passed as a argument

Button component with its own logic inside

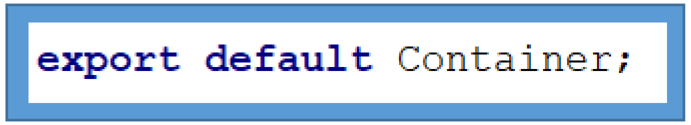
accenture

Demo – Higher Order Component (Cont …)



```
let NewButton=BaseComponent(Button);

const Label=()=>{
    return(
        <label>This is from Label</label>
    )
}
let NewLabel=BaseComponent(Label);
class Container extends Component{
    render(){
        return(
            <div>
                <NewButton/>
                <NewLabel/>
            </div>
        )
    }
}
```



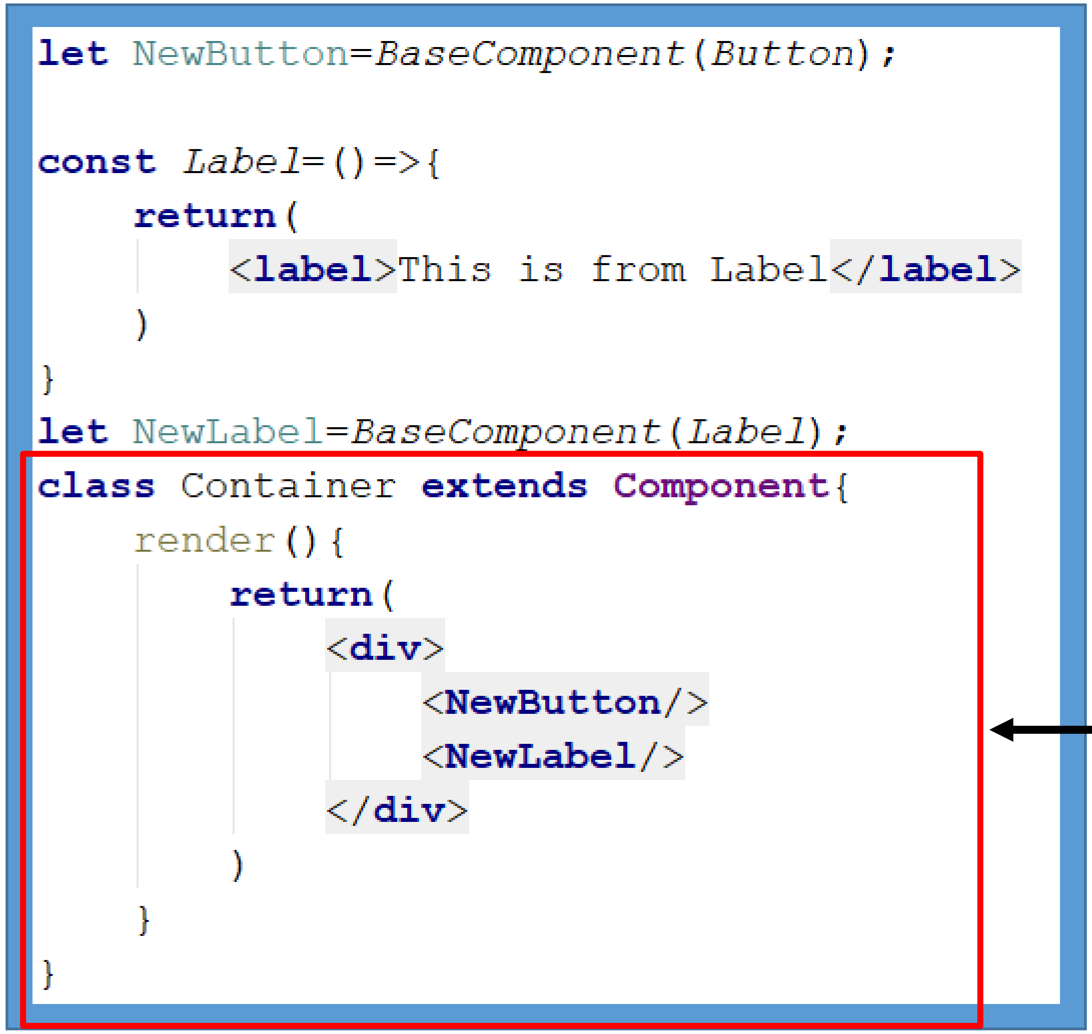
```
export default Container;
```

BaseComponent function is called by passing Button Component. The returned value is stored inside NewButton

Label component is passed to BaseComponent the returned value is stored NewLabel

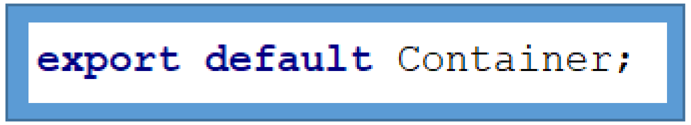
accenture

Demo – Higher Order Component (Cont ...)



```
let NewButton=BaseComponent(Button);

const Label=()=>{
    return(
        <label>This is from Label</label>
    )
}
let NewLabel=BaseComponent(Label);
class Container extends Component{
    render(){
        return(
            <div>
                <NewButton/>
                <NewLabel/>
            </div>
        )
    }
}
```

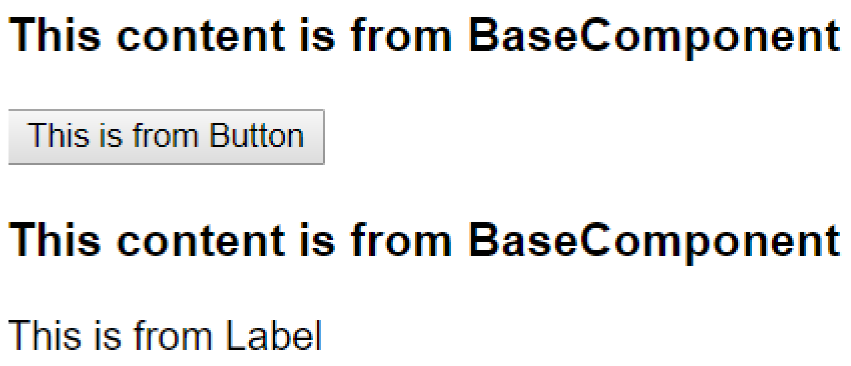


```
export default Container;
```

Finally Container is responsible to render NewButton and NewLabel.

Demo – Higher Order Component (Cont …)

Output



```
This content is from BaseComponent

[This is from Button]

This content is from BaseComponent

This is from Label
```

NewButton and NewLabel will display its own logic along with Base Component's logic

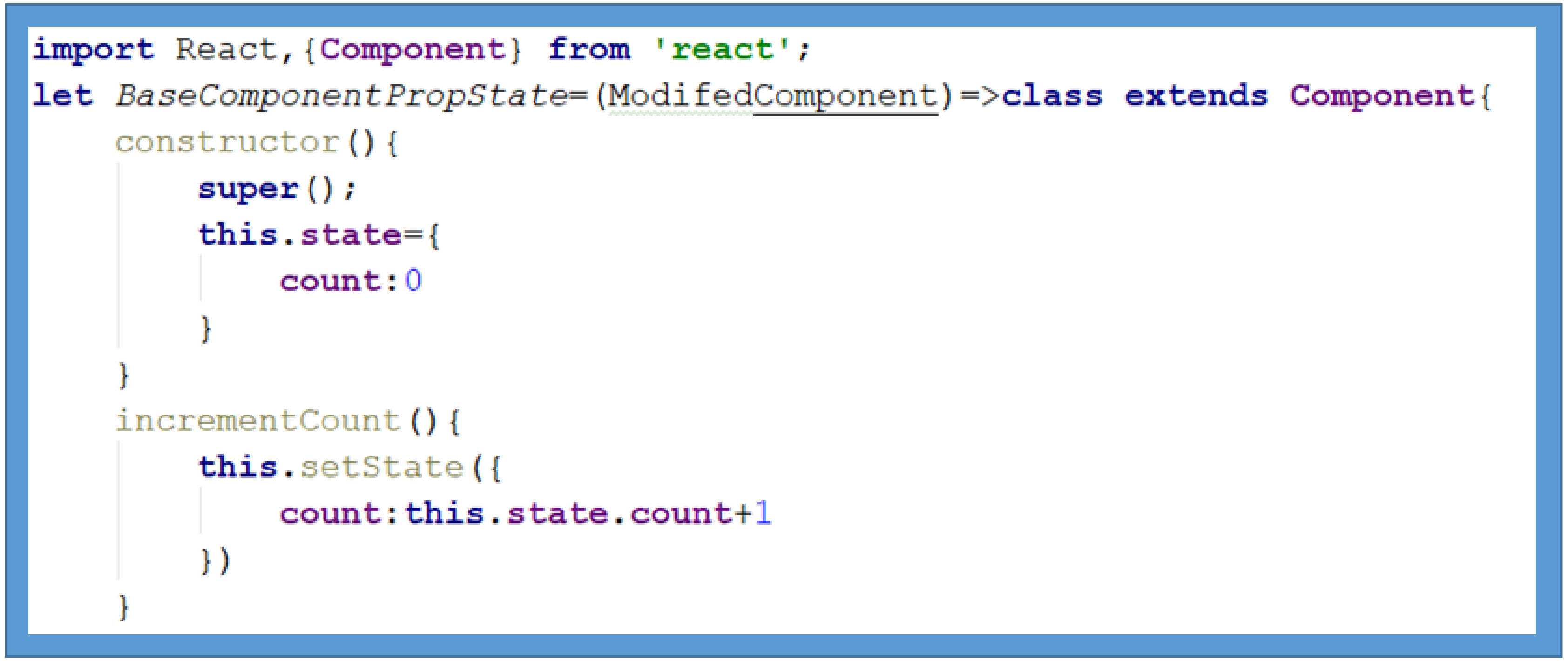
Higher Order Component – Prop and State

Higher order components can also have some state within it , which can be utilized by other component

When the BaseComponent have states, this will be provided to other component as a props

Demo – Higher Order Component with Prop and State

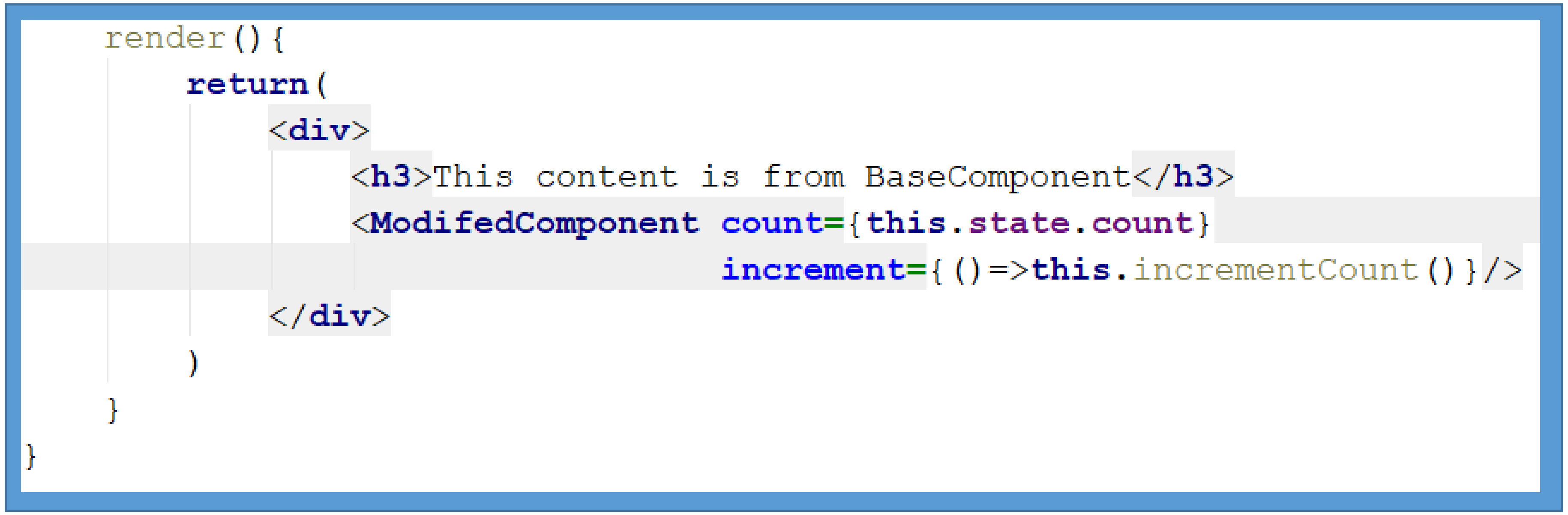
Create a new class BaseComponentPropState.js



```
import React,{Component} from 'react';
let BaseComponentPropState=(ModifedComponent)=>class extends Component{
    constructor(){
        super();
        this.state={
            count:0
        }
    }
    incrementCount(){
        this.setState({
            count:this.state.count+1
        })
    }
```

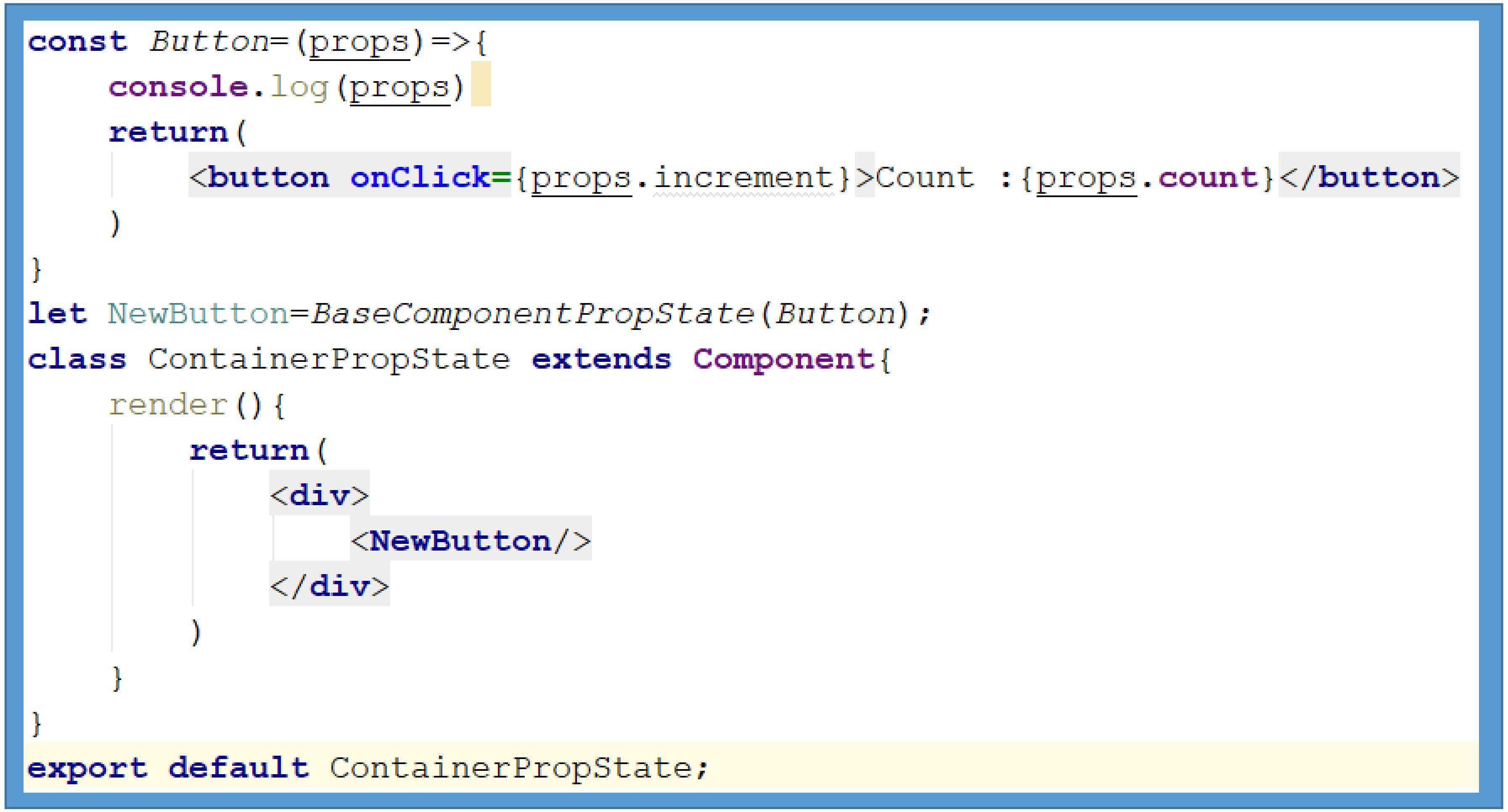
accenture

Demo – Higher Order Component with Prop and State (Cont…)



```
    render(){
        return(
            <div>
                <h3>This content is from BaseComponent</h3>
                <ModifedComponent count={this.state.count}
                                  increment={()=>this.incrementCount()}/>
            </div>
        )
    }
}
```

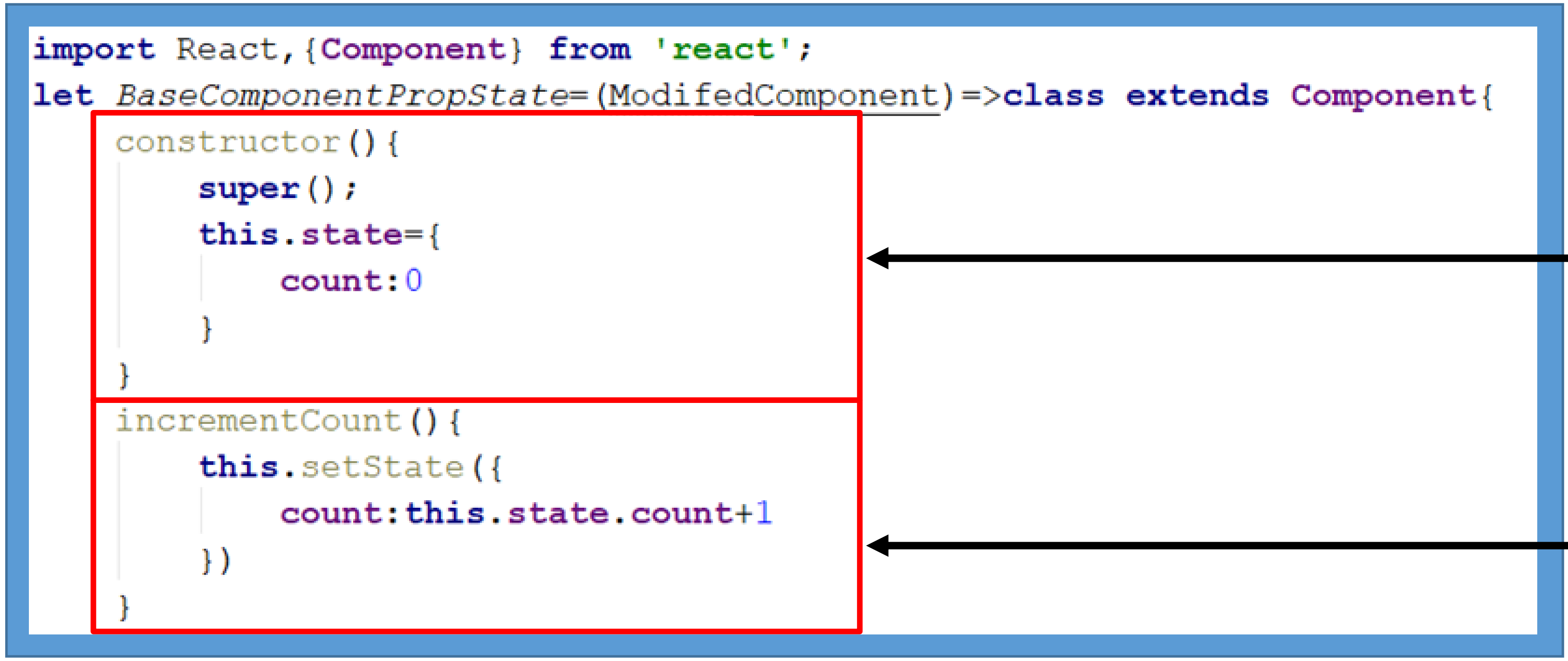
Demo – Higher Order Component with Prop and State (Cont…)



```
const Button=(props)=>{
    console.log(props)
    return(
        <button onClick={props.increment}>Count :{props.count}</button>
    )
}
let NewButton=BaseComponentPropState(Button);
class ContainerPropState extends Component{
    render(){
        return(
            <div>
                <NewButton/>
            </div>
        )
    }
}
export default ContainerPropState;
```

Demo – Higher Order Component with Prop and State (Cont…)

Create a new class BaseComponentPropState.js

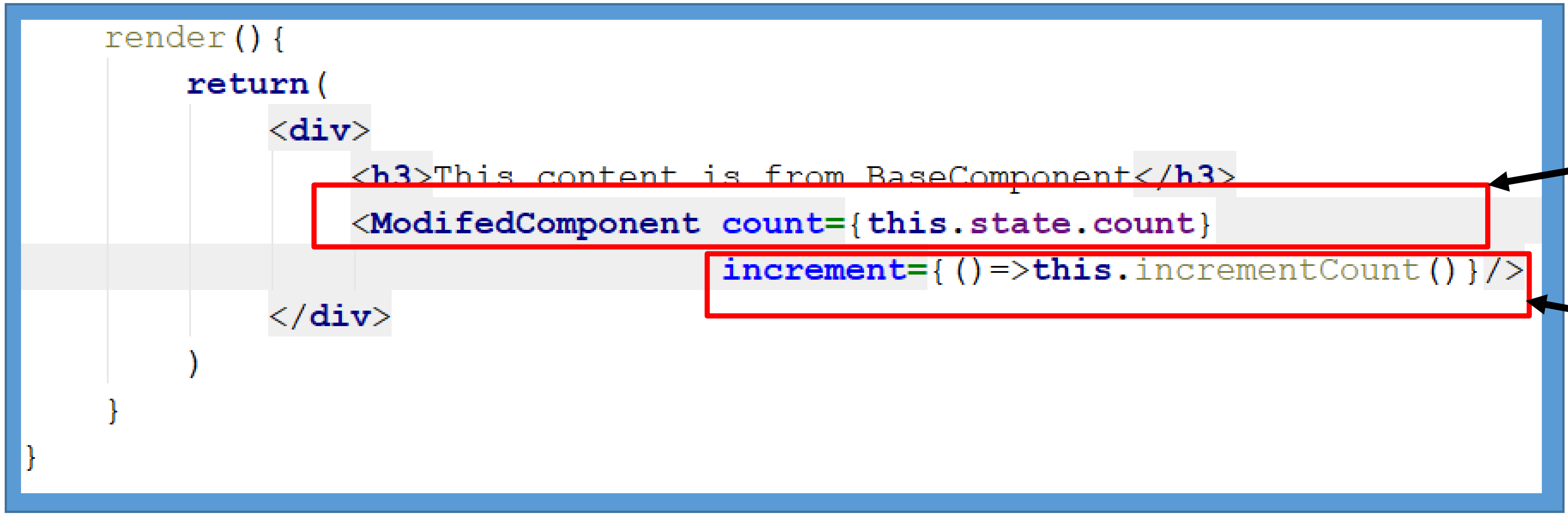


```
import React,{Component} from 'react';
let BaseComponentPropState=(ModifedComponent)=>class extends Component{
    constructor(){
        super();
        this.state={
            count:0
        }
    }
    incrementCount(){
        this.setState({
            count:this.state.count+1
        })
    }
```

Here “count” is a State of a BaseComponent, which has initial value as 0

This method will use setState to increase the value of a counter by 1

Demo – Higher Order Component with Prop and State (Cont…)



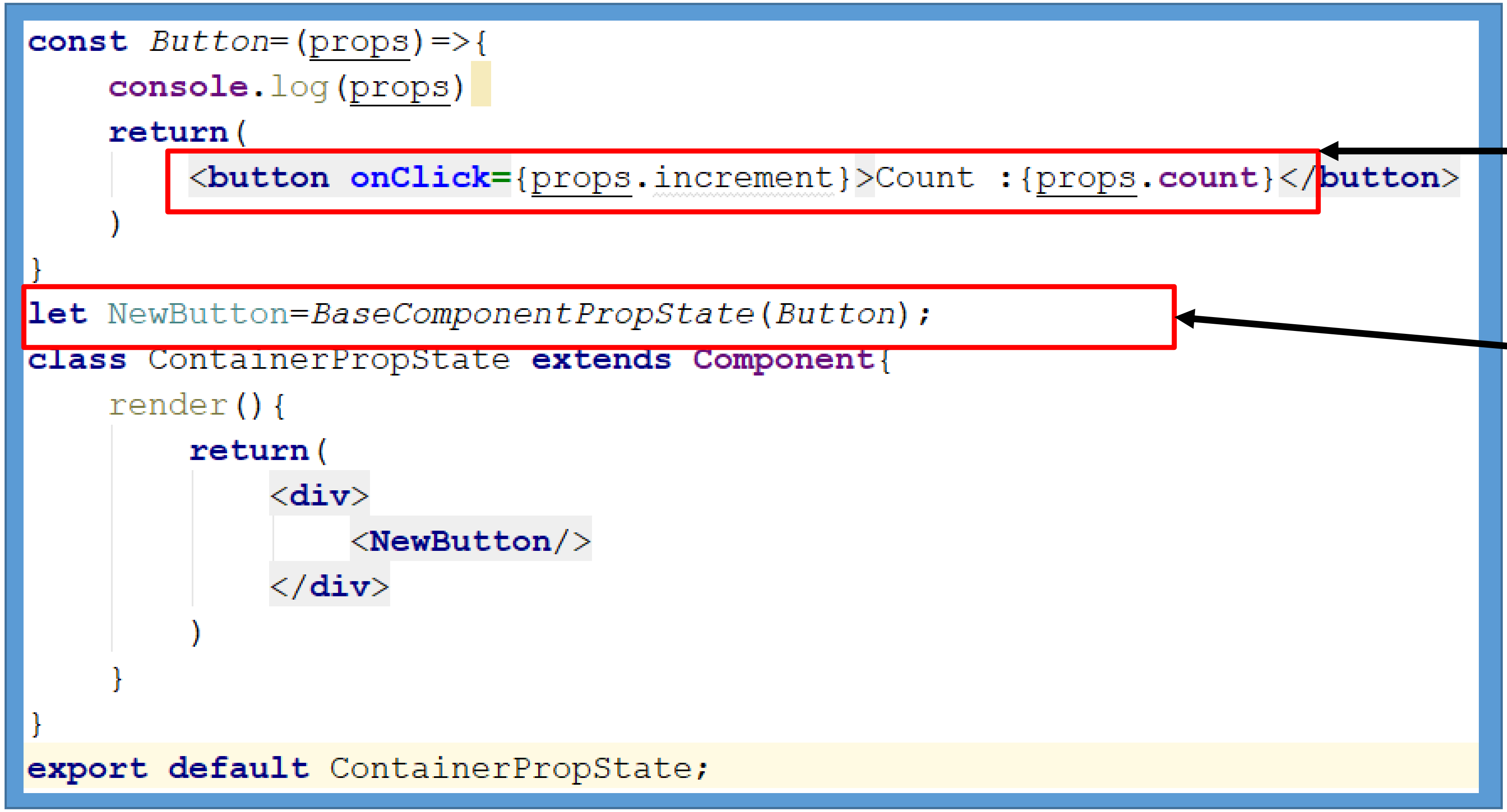
```
    render(){
        return(
            <div>
                <h3>This content is from BaseComponent</h3>
                <ModifedComponent count={this.state.count}
                                          increment={()=>this.incrementCount()}/>
            </div>
        )
    }
}
```

The current value of a count is captured

This will call incrementCount method which will increment value of a count

accenture

Demo – Higher Order Component with Prop and State (Cont…)

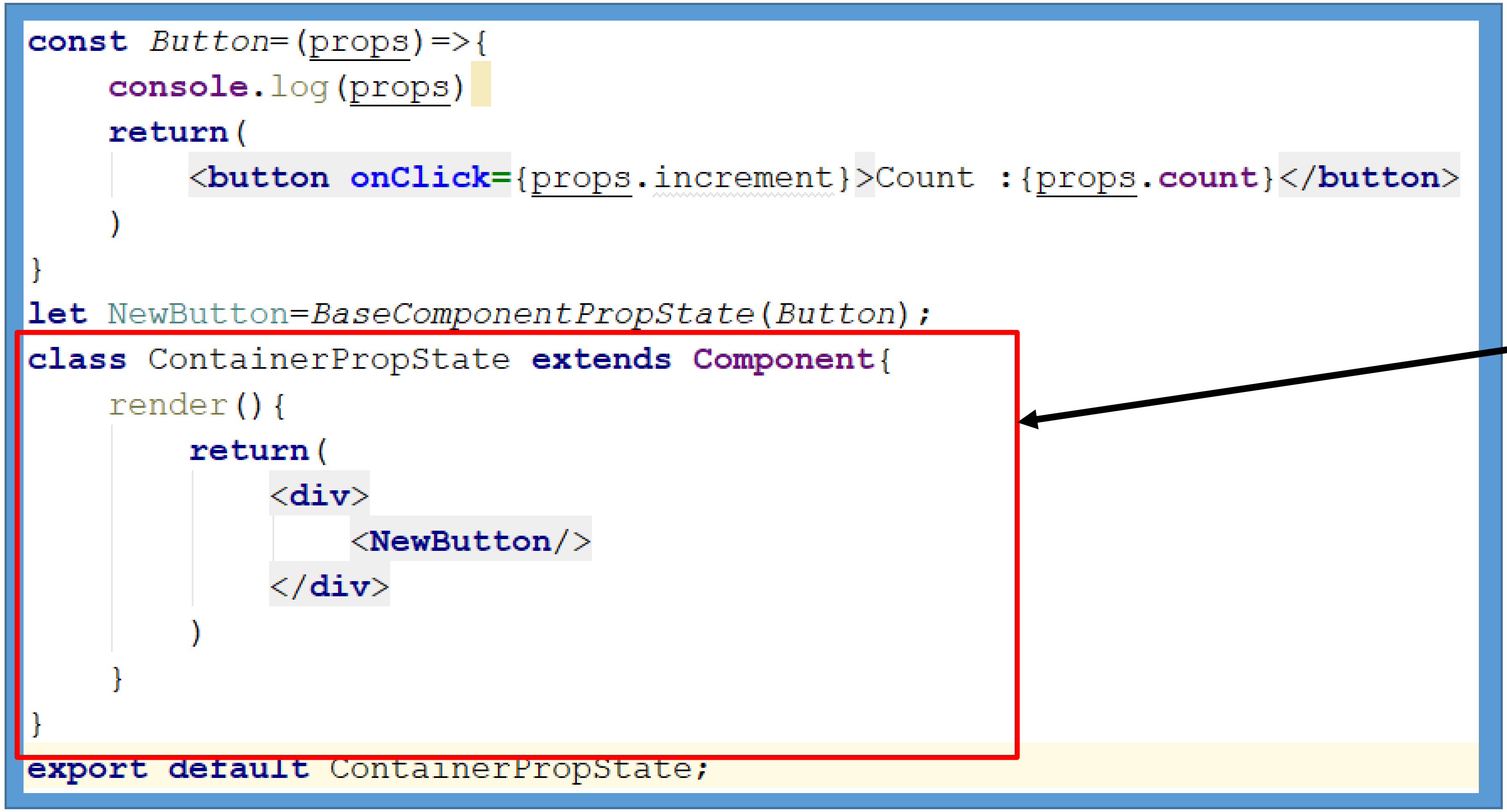


```
const Button=(props)=>{
    console.log(props)
    return(
        <button onClick={props.increment}>Count :{props.count}</button>
    )
}
let NewButton=BaseComponentPropState(Button);
class ContainerPropState extends Component{
    render(){
        return(
            <div>
                <NewButton/>
            </div>
        )
    }
}
export default ContainerPropState;
```

Button will have props to take values from other component, here it will take the props from BaseComponent

Button component is passed as a parameter to BaseComponent which will get complete state from BaseComponent

Demo – Higher Order Component with Prop and State (Cont…)



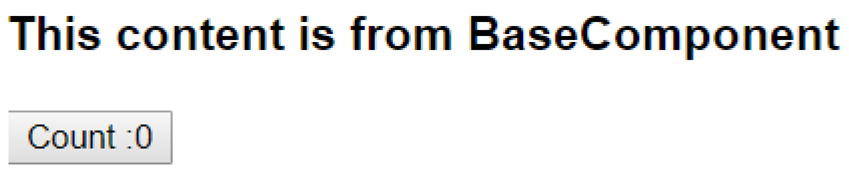
```
const Button=(props)=>{
    console.log(props)
    return(
        <button onClick={props.increment}>Count :{props.count}</button>
    )
}
let NewButton=BaseComponentPropState(Button);
class ContainerPropState extends Component{
    render(){
        return(
            <div>
                <NewButton/>
            </div>
        )
    }
}
export default ContainerPropState;
```

Finally ContainerPropState is rendered to display output

accenture

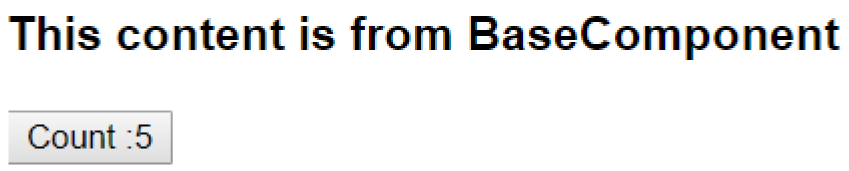
Demo – Higher Order Component with Prop and State (Cont…)

Output



This content is from BaseComponent

Click on the count button to increment the value



This content is from BaseComponent

Activity

1. Create a higher order component which will have state as password and needpassword

2. This HOC can decide whether to send the password to component or not

3. If the state(needpassword) is set to yes then the component will receive password and display output message in a button as “I got password” along with the message password should be displayed

4. If the state(needPassword) is set to no then the component will not receive password and display output message in a button as “I have not received password”

MODULE SUMMARY

- Understand what is higher order component
- What are the benefits of HOC
- Implement HOC

accenture

THANK YOU

accenture