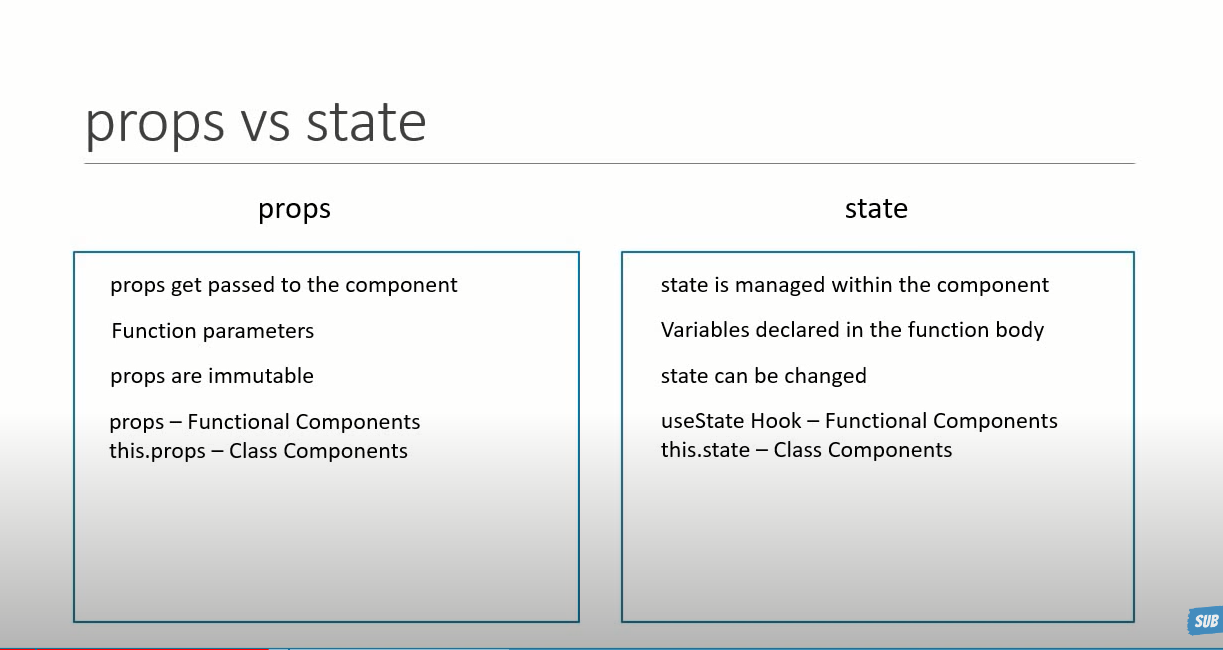
State React :

* Prop is immutable type(we can’t change the value after assign once) so other option is component state.
* For eg you are showing message welcome something and after clicking some button you have to show another message so state comes in picture.
* State is nothing but an object privately maintained inside a component.
* State can influence what is rendered in the browser.
* State can be changed with in component.
* If you are going to set state directly without “SetState” then aur value will be set but that changes not going to reflect in Ui.
* Never modified state directly other wise that value not render in component



Step 1 : Create state object initialize. This thing done mostly in constructor.

//step 1 : Create state object initialize. This thing done mostly in constructor.

constructor() {

    super()

this.state={

    message: 'Welcome visitor'

}

}

Super() in React component

Super() function is to call the constructor of the parent class. It is used when we need to access a few variables in the parent class. It returns an object which represents the parent class. The right way to use it is when the child class and parent class are from the same field.

Step 2: call the state

    render (){

        return (

            <div>

                {/\* step 2 : calll the state \*/}

                <h1>{this.state.message}</h1>

            </div>

        )

    }

Step 3 : add click event for change message

                {/\* step 3 : add click event for change message  \*/}

               {/\*()=> this.changeMessage() = we have created handler  \*/}

                <button onClick={()=> this.changeMessage()}> Subscribe</button>

Step 4: Create function to set state

 //step 4 : Create function for set state

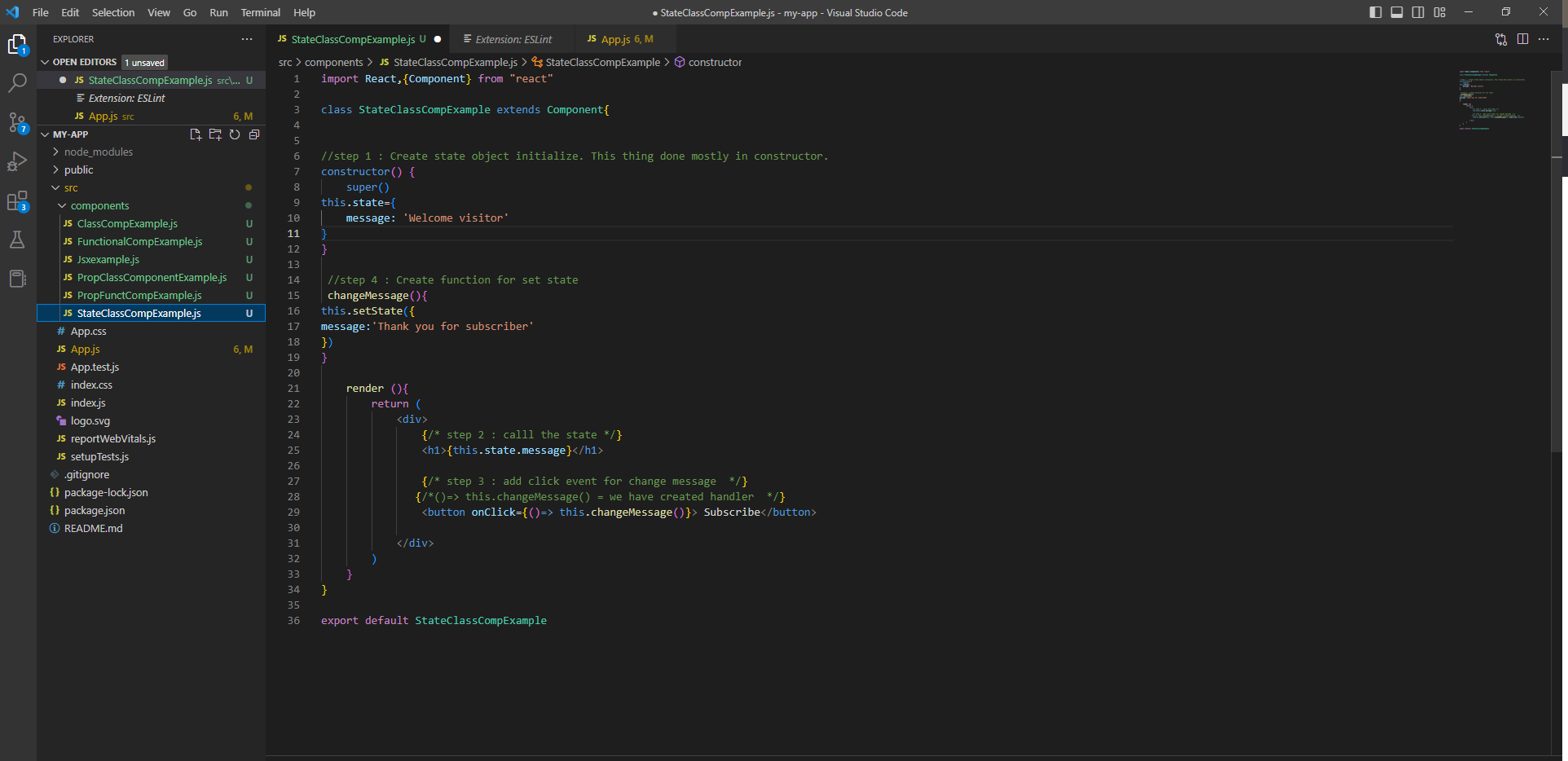
 changeMessage(){

this.setState({

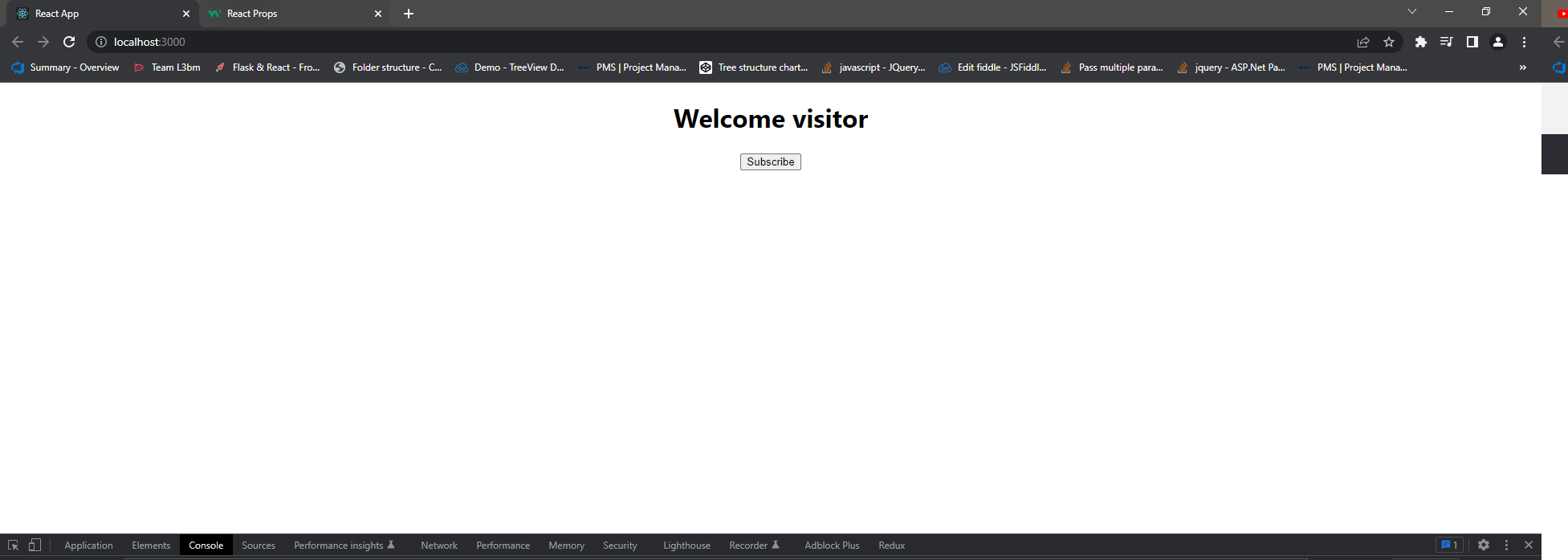
message:'Thank you for subscriber'

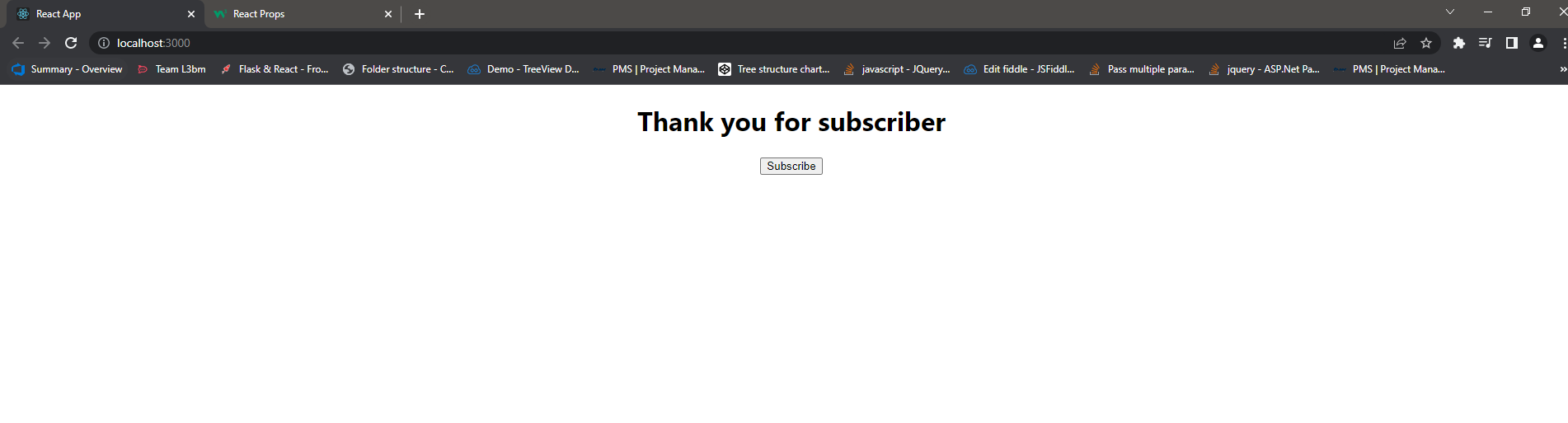
})

}

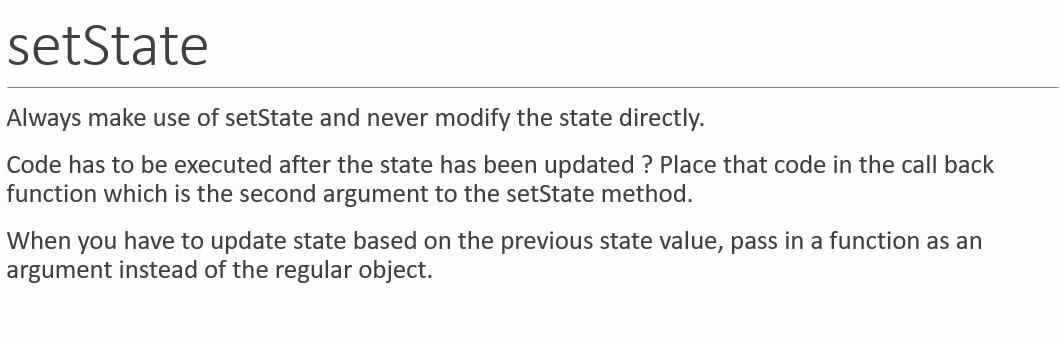


Output :





Set State :



* Never modified state directly other wise that value not render in component
* Set state have two parameter

1. State object
2. Callback function

increment(){

        this.setState({

            count:this.state.count+1

        },

        ()=>{

            console.log('Callback value',this.state.count)

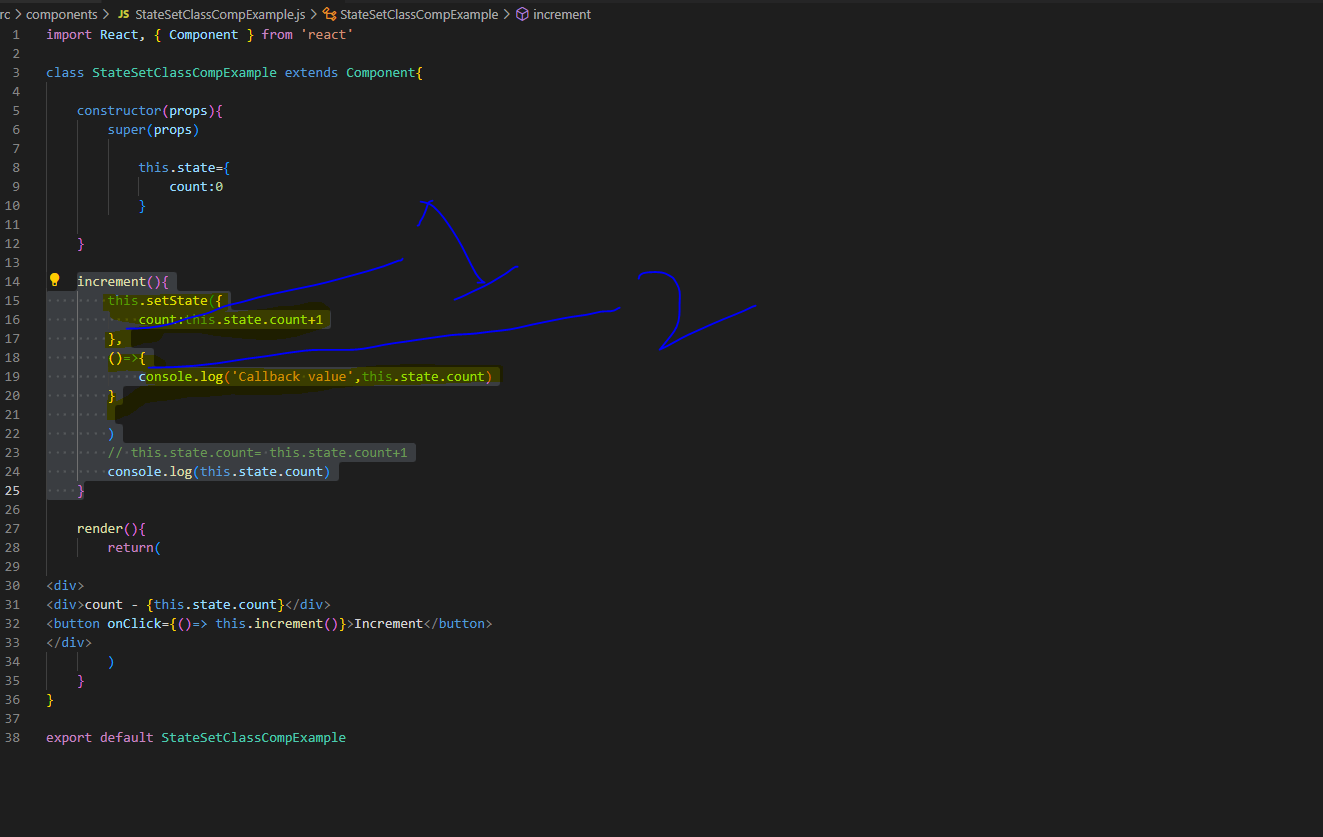
        }

        )

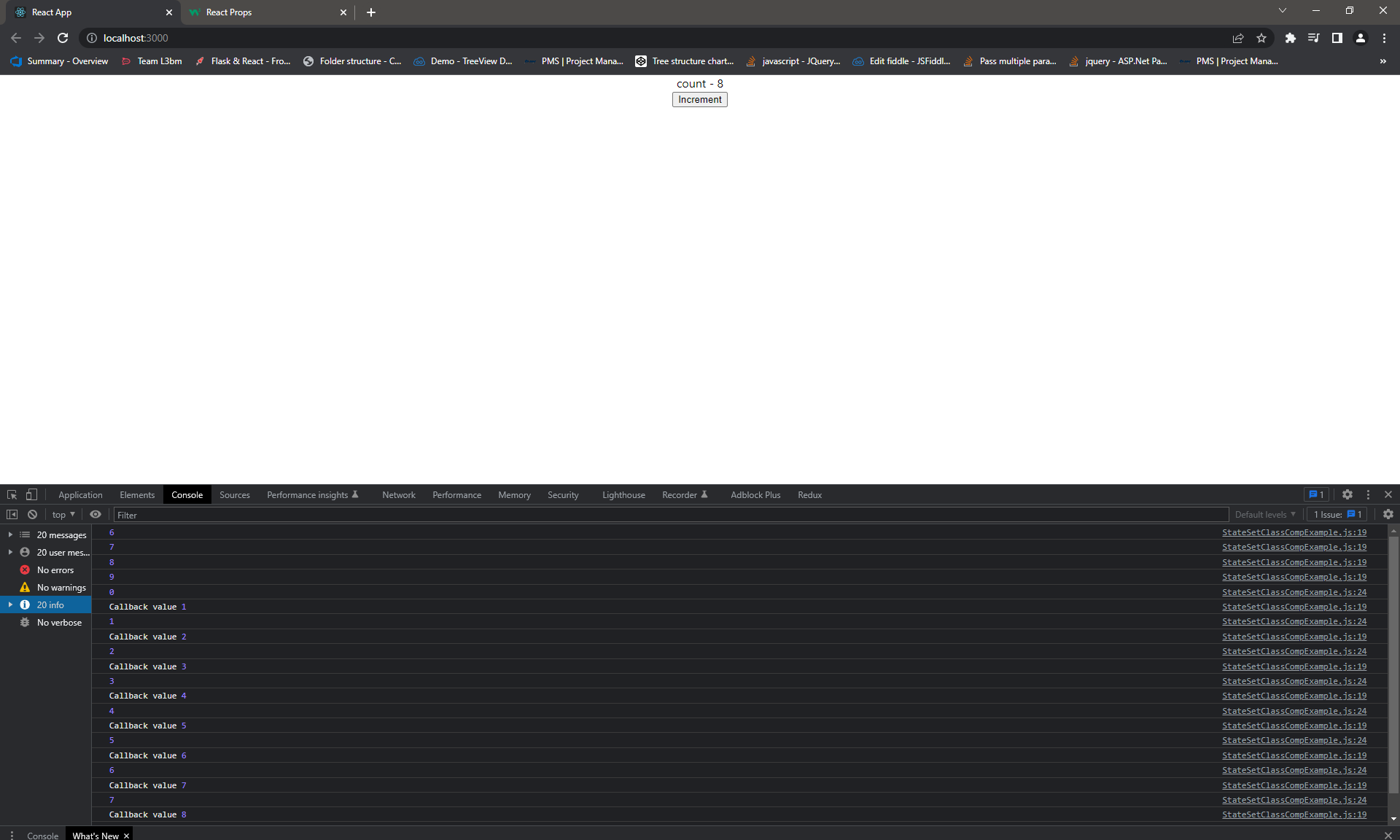
        // this.state.count= this.state.count+1

        console.log(this.state.count)

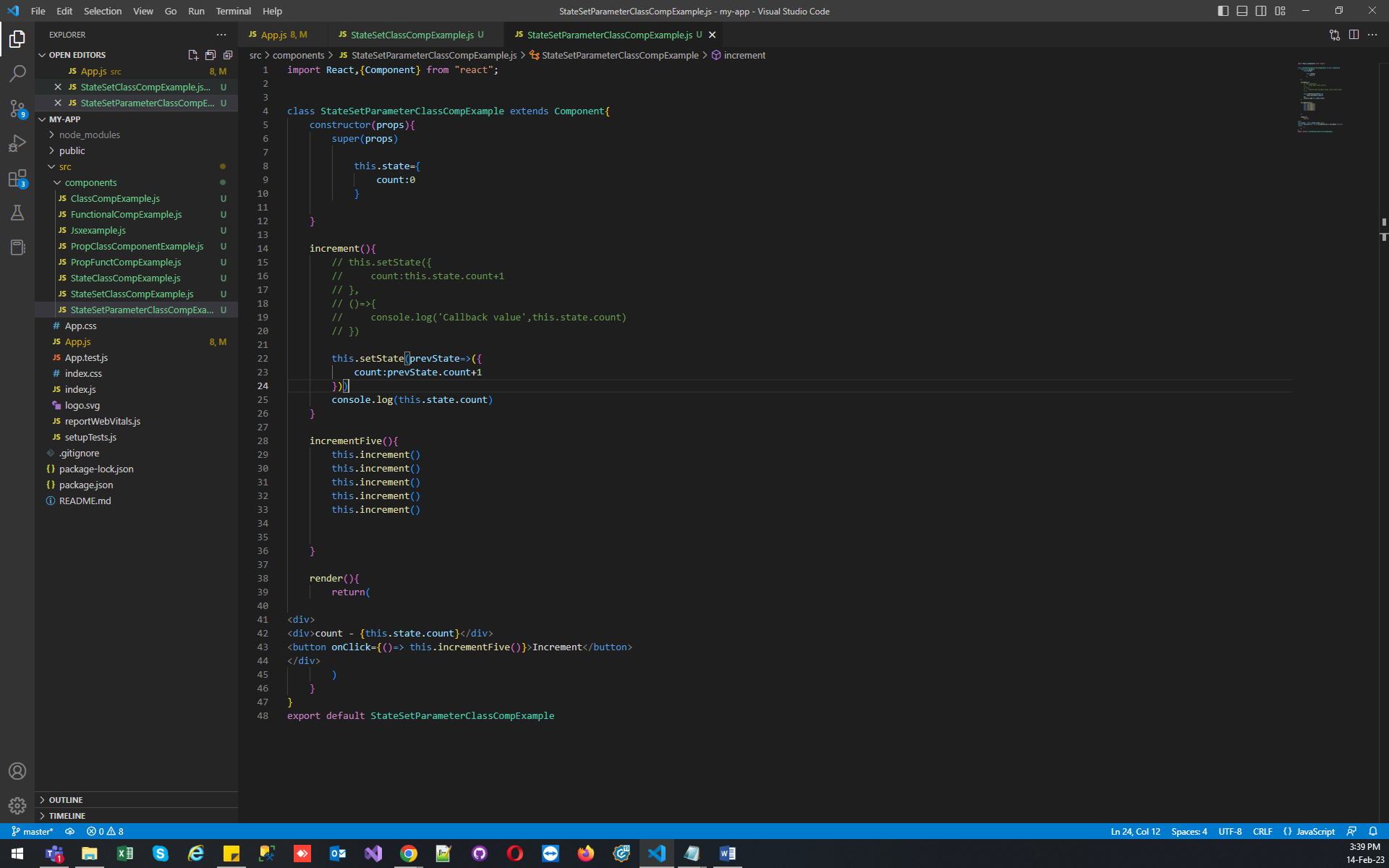
    }

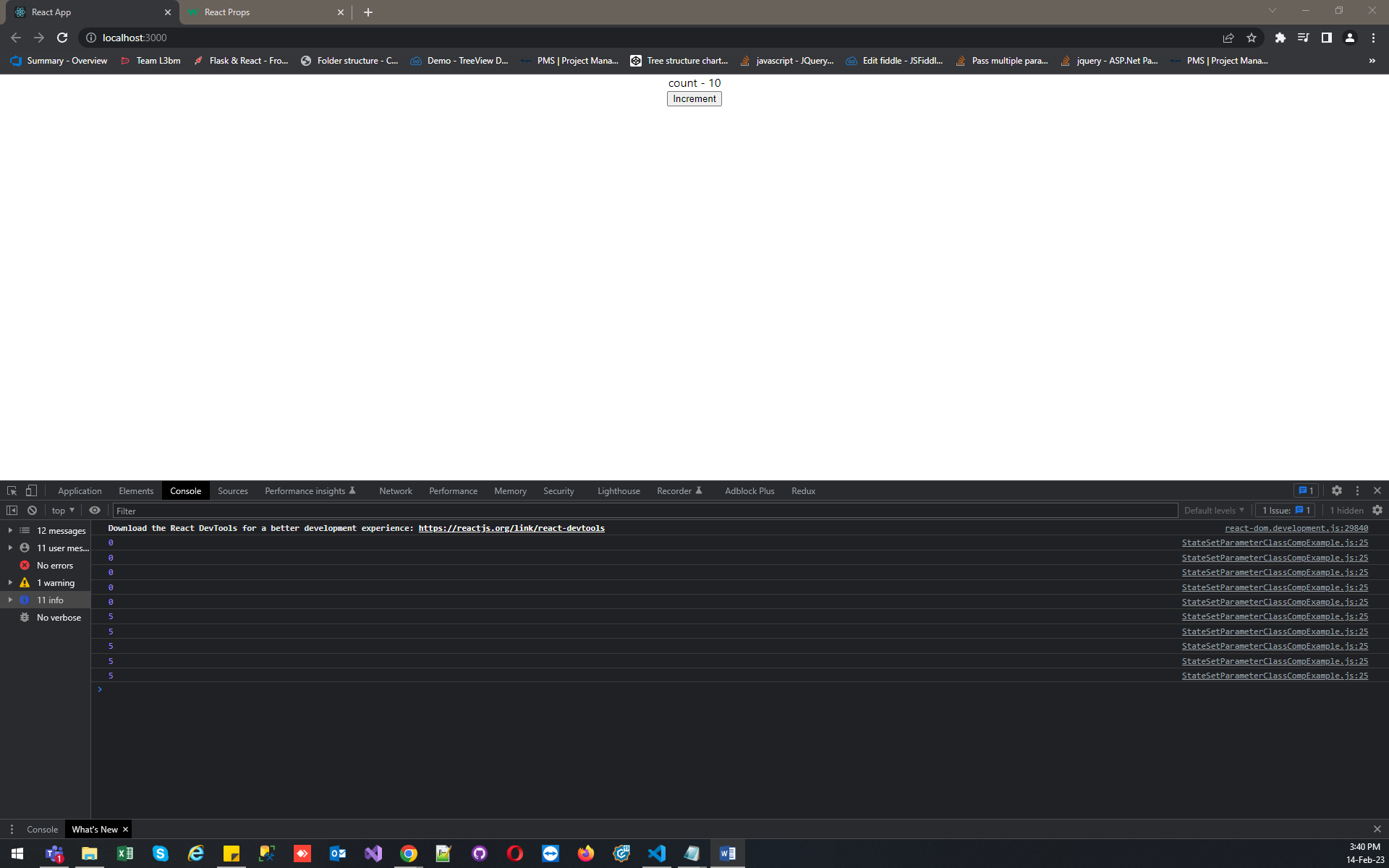


Output :



If you want to update state based on previous state then make sure pass function as a argument regular object. Function can be access previous state for calculate.





  this.setState((prevState, props)=>({

            count:prevState.count+1

        }))