NOTE: imp\*  
>Do not mutate/change state variable directly, use setState instead.  
  
Class Based component:  
>class based component is a normal js class which extends React.component class and it has render() method which return some piece of JSX, which later convert into html using babel and render it on browser .  
  
  
=>syntax to create class component:

class profile extend React.component

{

Render(){

}

}  
  
>’extend’ key is use to let know React that it’s not a normal js class instead it’s a component.  
>‘extend’ also use to inherit sum properties from React component into class component (i.e, ‘profile’)  
  
>render() method is most imp. Part of class based comp, , we can’t create class without it, .  
>render() => return=>JSX ,which is basically render into dom  
>’this’ keyword mostly use in class based component.  
  
  
>when props given to a component react attach it with class based comp. that’s y it’s accessible by ‘this’ keyword,React track the comp..  
  
//constructor:  
>It is use to initialization,   
>It is best place to create State and receive props.  
>whenever we load a class comp. Constructor invoke once initially before render().   
>when instance of class created , constructor calls first and all the passed props get attached with constructor props which later accessible by ‘this.props’ keywords.  
>All the passed props is combined in a single object and that object will be passed to constructor parameter(i.e props).  
  
Constructor(props)

{

Super(props)

// we can create state variable here

}  
  
>in class based comp. all the statevariable are created as a part of one object.  
>React use one big object to maintain all the state.  
  
>syntax:  
  
constructor(props){

super(props)

this.state={

var1:123,

count:0

}

}  
  
//this.setState()

this.setState({

//     count:this.state.count+1,

//     var1:"ram"

//     })

BTS, It only updates the passed state variables, it doesn’t touch values of other state variable.  
>always update state varaibles using setState() method never update it directly .

e.g ,like this

{/\* {this.state.count=1}  \*\*\*never do that, never mutate state directly\*/}

Imp: whenever a class based comp. load in web-page ,a life cycle method call.  
  
It happens in two phase:

1>render phase:

2>commit phase  
  
>render phase-> where constructor(), render() calls

>commit phase-> where react actual update the DOM ,then componentDidMount calls.  
  
when a class load/mount first constructor is called, once constructor call than second render(), and third compDidMount() call  
>calling orders are:  
1>constructor()

2>render()

3>componentDidMount()

//componentDidMount() :  
>It’s a best place for Api calling.  
Q why API call in CompDidMount()?  
Ans. Bcoz od rendering phase, and to make app fast we got follow this principle  
Render comp=>make API call=>update data=>re-render the comp.  
  
Render phase:all reconciliation process happens here,Everything in render phase is happening in virtual Dom(as it’s fast, bcoz it deals with object)  
  
>React batching multiple children bcoz once commit phase start React try to update the DOM & Dom manipulation is most expensive when we update/load component as it takes lot time,   
  
>render phase is also fast than commit phase bcoz for optimization it batch all child class comp at render phase so that Dom update happens in single batch as commit phase takes lot time than render phase, also DOM manipulation/updation is expensive .  
  
  
>React batch the render phase of all children and after achiving render phase by multiple children than commit phase happen  
  
mounting:it means showing UI on page.  
unmounting :it means removing component/UI from page.  
  
componentDidUpdate():It calls after API call and if there is any state update ,dom get updated by new data, than componentDidUpdate call.  
>It calls after every update, similar to passing a state dependency in useEffect.  
  
  
>If there is no props or state update , zero times componentDidUpdate() call.

componentDidMount(){  
//need to clear this intervel otherwise it will keep callingthis.timer= setInterval(()=>{console.log("extra compDidMount")},1000)}  
  
componentDidUpdate(){

console.log("extra compDidUpdate")

}

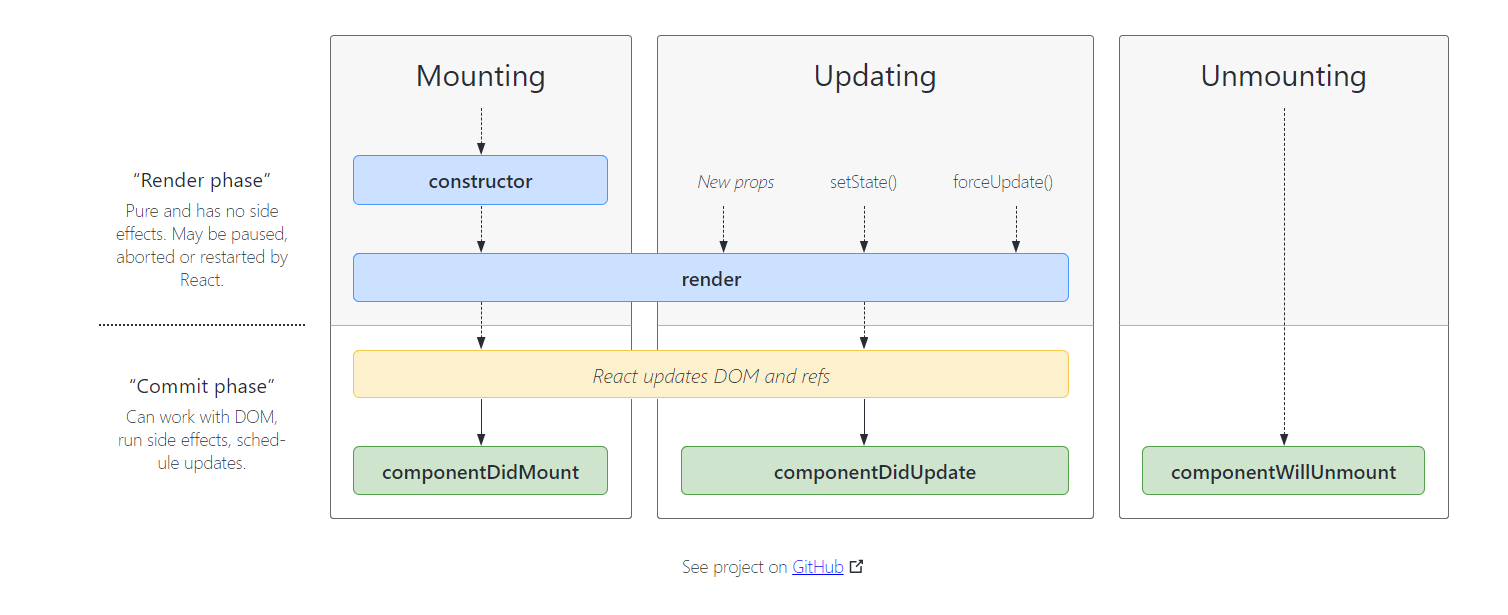
componentWillUnmount(){

//clearing interval when switching one comp to another.

clearInterval(this.timer)

console.log("extra compWillUnmount")

}

componetWillUnmount():it calls when component get changed in the page.  
// lots of thing need to clear before while unmounting/remove component from page ,like clear intervals  
  


//Q. why do we use super(props)?  
//Q. why can’t we use async with useEffect callback func but we can use async with componetDidMount?