Class based components

First lets see how we have created the functional components:

const Users = (props) =>{

return(

<div className="card-container">

<h2>Name: {props.name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

</div>

)

}

export default Users

Now see same thing with class based components

import React from "react";

class UsersClass extends React.Component {

constructor(props) {

super(props)

}

render(){

// we can also do like

const {name, location} = this.props;

return(

<div className="card-container">

{/\* <h2>Name: {this.props.name}</h2> \*/}

<h2>Name: {name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

</div>

)

}

}

export default UsersClass;

How to use State here

1st see in functional components

import { useState } from "react"

const Users = (props) =>{

const [count, setCount] = useState(0)

return(

<div className="card-container">

<h2>Name: {props.name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

<h4>Count : {count}</h4>

</div>

)

}

export default Users

Now in class

import React from "react";

class UsersClass extends React.Component {

constructor(props) {

super(props);

this.state = {

count: 0

}

}

render(){

// we can also do like

const {name, location} = this.props;

return(

<div className="card-container">

{/\* <h2>Name: {this.props.name}</h2> \*/}

<h2>Name: {name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

<h4>Count: {this.state.count}</h4>

</div>

)

}

}

export default UsersClass;

We can destructure also like

Const { count } = this.state;

{count} /// will print value of count

In class

We can mention all state variable in one state object

This.state = {

One:1

Two:2

…

}

How to update state variable in class

NEVER UPDATE STATE variable DIRECTLY.

~~This.state.count = this.state.count + 1;~~

We can use this.setState method to update it

this.setState = {

Count: this.state.count+1;

}

this.setState = {

Count: this.state.count+1,

Count2: this.state.count+1,

}

We can update the state or more then one variable at once with help of this.setState method of class.

Function

const [count, setCount] = useState(0);

const [count2, setCount2] = useState(0);

Class

constructor(props) {

super(props);

this.state = {

count: 0,

count2: 0,

}

}

How to update state valiables?

Part 3

How to update state variables ?

Functional:

import { useState } from "react"

const Users = (props) =>{

const [count, setCount] = useState(0);

const [count2, setCount2] = useState(0);

const updateCounter = () =>{

setCount(count+1);

}

return(

<div className="card-container">

<h2>Name: {props.name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

<h4>Count : {count}</h4>

<h4>Count 2 : {count2}</h4>

<button onClick={updateCounter}>+1</button>

</div>

)

}

export default Users

Class

import React from "react";

class UsersClass extends React.Component {

constructor(props) {

super(props);

this.state = {

count: 0,

count2: 0,

}

}

render(){

// we can also do like

const {name, location} = this.props;

return(

<div className="card-container">

{/\* <h2>Name: {this.props.name}</h2> \*/}

<h2>Name: {name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

<h4>Count: {this.state.count}</h4>

<button onClick={()=>{

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

// NEVER UPDATE VIRABLES DIRECTLY

this.setState({

count : this.state.count + 1

})

}}>+1</button>

</div>

)

}

}

export default UsersClass;

There is function called componentDidMount()

Which is called once your component is created

import React from "react";

class UsersClass extends React.Component {

constructor(props) {

super(props);

console.log("Constructor involed");

this.state = {

count: 0,

count2: 0,

}

}

componentDidMount(){

console.log("Component Did Mount");

}

render(){

console.log('Render method invoked');

// we can also do like

const {name, location} = this.props;

return(

<div className="card-container">

{/\* <h2>Name: {this.props.name}</h2> \*/}

<h2>Name: {name}</h2>

<h3>Location: ABC</h3>

<h4>Contact Details</h4>

<h4>Count: {this.state.count}</h4>

<button onClick={()=>{

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

// NEVER UPDATE VIRABLES DIRECTLY

this.setState({

count : this.state.count + 1

})

}}>+1</button>

</div>

)

}

}

export default UsersClass;

Output

Constructor involved

Render method invoked

Component Did Mount

Like useEffect in functional programming,

We call apis in compoenentDidMount Method.

Part 5

How to make API call?

<https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

React have 2 phase

1. Render
2. Commit

First it will construct the things

Then it will render the data

Then ir will update the dom

Then it ill call component DId Mount mothod.

As per above link we will get diagram based on that diagram we can understand the flow

Code:

import { Component } from "react";

import UsersClass from "./UsersClass";

class About extends Component {

constructor(props){

super(props);

console.log("About us Parent Constructor Called");

}

componentDidMount(){

console.log("Parent Component Did Mount method Called");

}

render(){

console.log("Parent Render Method Called");

return(

<div>

<h1> ABout us Class Component</h1>

<UsersClass name={'first'} location={'First'} />

<UsersClass name={'Second'} location={'second'} />

<UsersClass name={'Thirt'} location={'Thirt'} />

</div>

)

}

}

export default About;

Flow is

Parent Constructor Called

Parent Render Method Called

firstConstructor involed

firstRender method invoked

SecondConstructor involed

SecondRender method invoked

ThirtConstructor involed

ThirtRender method invoked

firstComponent Did Mount

SecondComponent Did Mount

ThirtComponent Did Mount

Parent Component Did Mount method Called

React has optimize here, it batches all the childs and then at last it will call ther componentDid mount method fo all together

Rendering the COm is very expensive work

So once start committing befire that it will complete all renders

**Commits takes time.**

0.

Part 6:

Steps: Life Cycle

—> Mounting

constructor (with Dummy data)

Render with Dumy data

Html loaded with Dymmy data

Component did mount called

Api called

setState with API data

— > Update

Render (with New APi data)

Html with new API data

Component did Update Called

There is one more method

Component will unmount

This will call when the component will disappear from the page

Like either we delete or we change the page.

In useEffect if we pass [] then it will called once

If we pass any value then on change of its state useEffect will trigger

In class-based

componentDidMount is called once only for mounting

From next time we have to call componentDidUpdate to update the values

It have 2 parameters prevProps, prevState

In useEffect we can call n number of times if we have different conditions for different state variable

In class-based we have to put everything in componentDidupdate with if else conditions.

Use of componenentWillUnmount

Support you created a setInterval function to call on every 5 seconds

And we know we are working on Single page application SPA

So if any component get updated still that setInterval function may run.

So if suppoe on speficid component we want to use the functionality which may affect the whole productivity

In that case we have to use componentwillunmount method to close them.

Even if we load same component again, it will call twice (n number of times we loaded)

Example;

import { Component } from "react";

import UsersClass from "./UsersClass";

class About extends Component {

constructor(props){

super(props);

console.log("About us Parent Constructor Called");

}

componentDidMount(){

console.log("Parent Component Did Mount method Called");

setInterval(() => {

console.log("set interval functionality");

}, 1000);

}

render(){

console.log("Parent Render Method Called");

return(

<div>

<h1> ABout us Class Component</h1>

<UsersClass name={'first'} location={'First'} />

{/\* <UsersClass name={'Second'} location={'second'} /> \*/}

{/\* <UsersClass name={'Thirt'} location={'Thirt'} /> \*/}

</div>

)

}

}

export default About;

This will create issue

We need to fix

Example:

import { Component } from "react";

import UsersClass from "./UsersClass";

class About extends Component {

constructor(props){

super(props);

console.log("About us Parent Constructor Called");

}

componentDidMount(){

console.log("Parent Component Did Mount method Called");

this.timer = setInterval(() => {

console.log("set interval functionality");

}, 1000);

}

componentWillUnmount(){

clearInterval(this.timer);

}

render(){

console.log("Parent Render Method Called");

return(

<div>

<h1> ABout us Class Component</h1>

<UsersClass name={'first'} location={'First'} />

{/\* <UsersClass name={'Second'} location={'second'} /> \*/}

{/\* <UsersClass name={'Thirt'} location={'Thirt'} /> \*/}

</div>

)

}

}

export default About;

What if we pass same thing setInterval in useEffect?

It will stop automatic?

No we didnt clear out mess.

useEffect have feature

It will return a function which will called when we will unmount it.