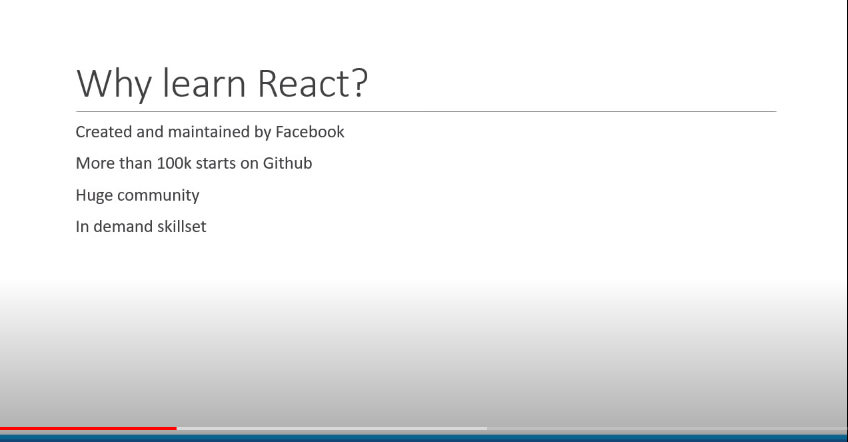
Toggle singleline comment out : Ctrl + / (// vice versa for commetn or uncomment)

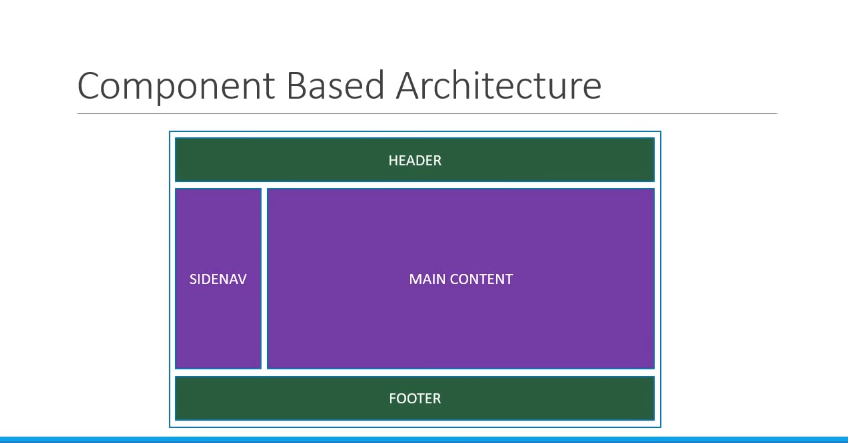
Togglemultipleline comment : Shift + Alt + A(/\* \*/Vice Versa for comment uncomment)

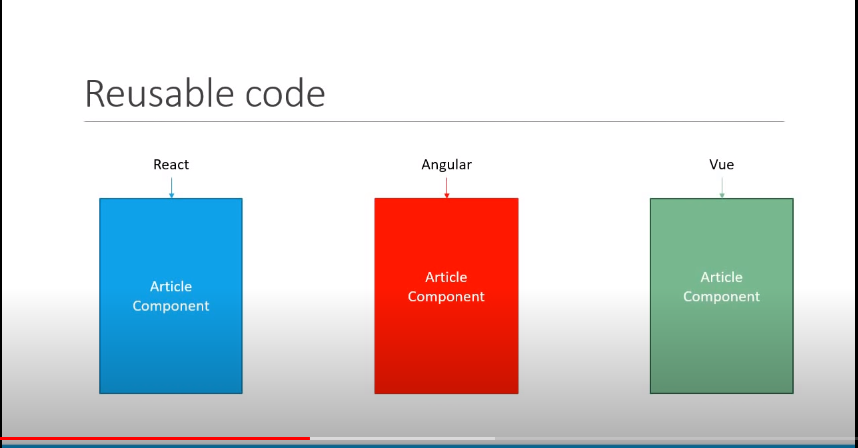
**Reactjs code snippets**Need

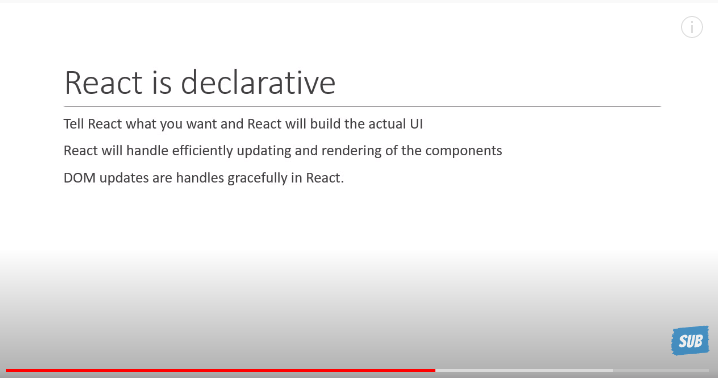
Need to install abv snippit for writing code in easy way

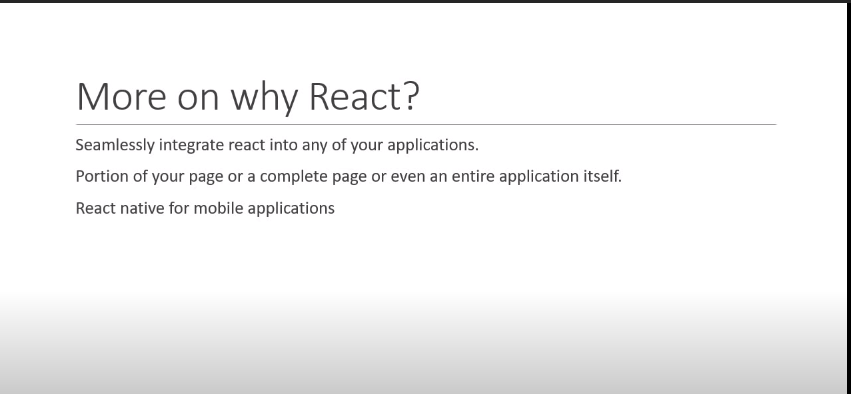


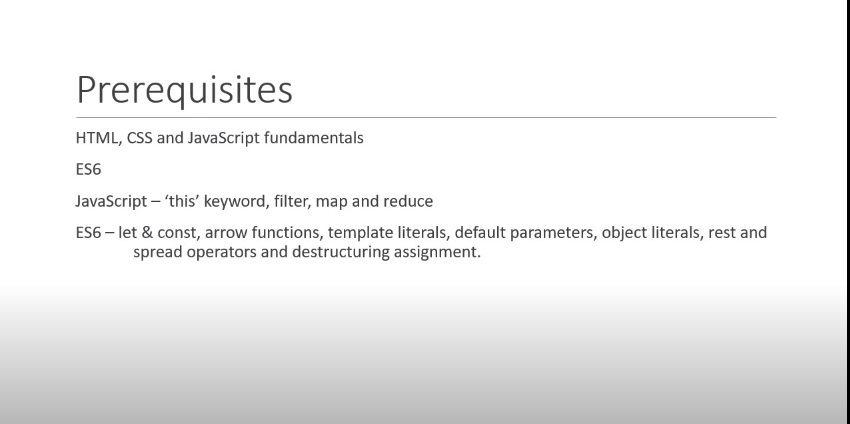


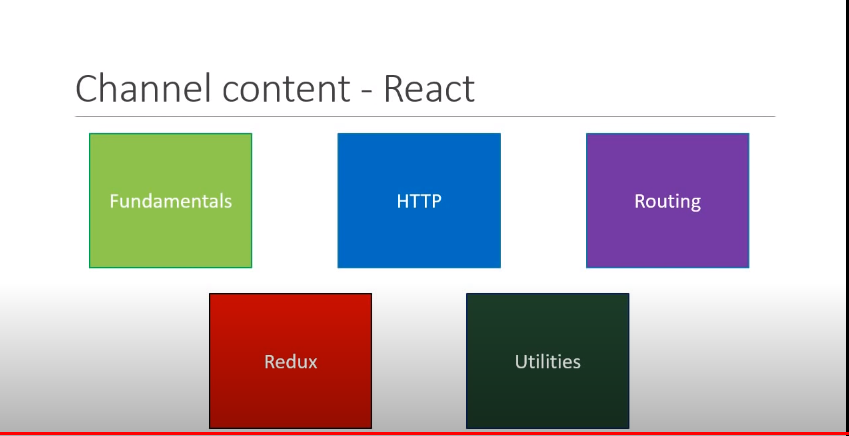


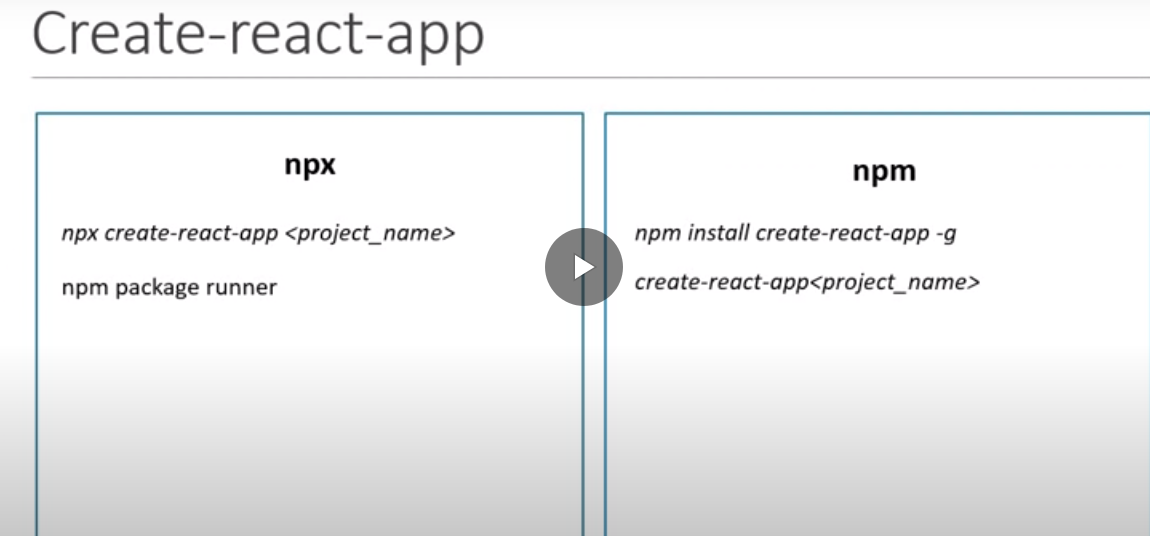






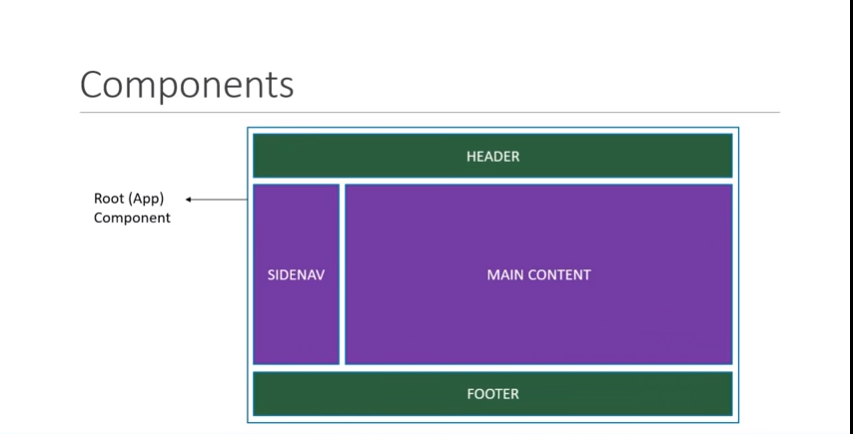


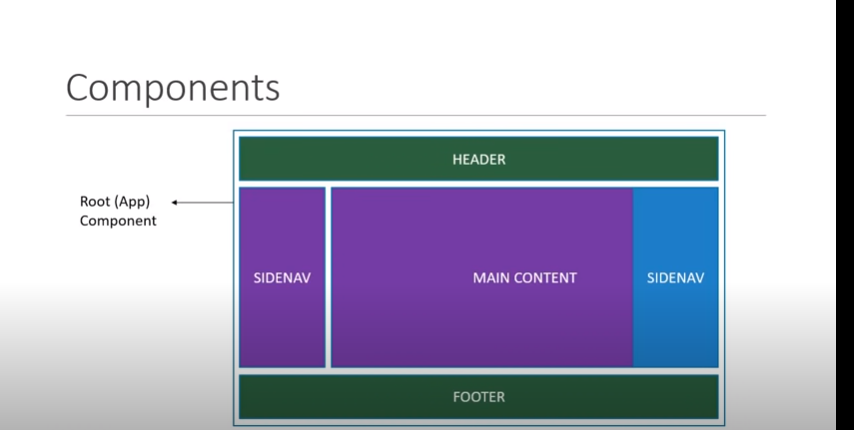


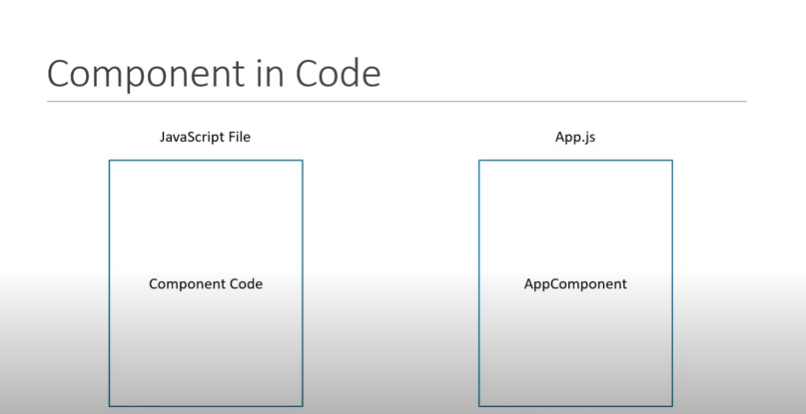


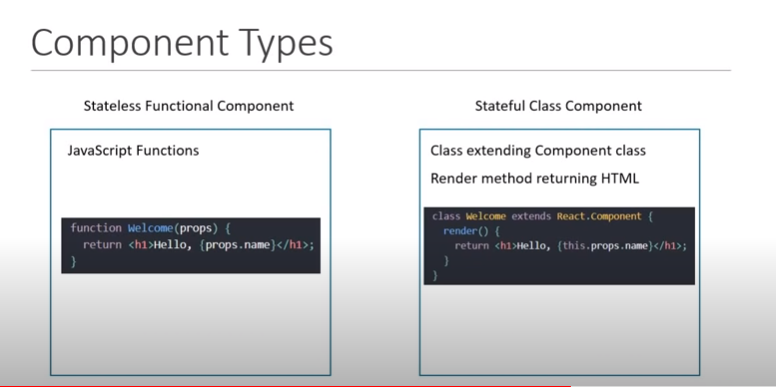
Folder Structure

Components:

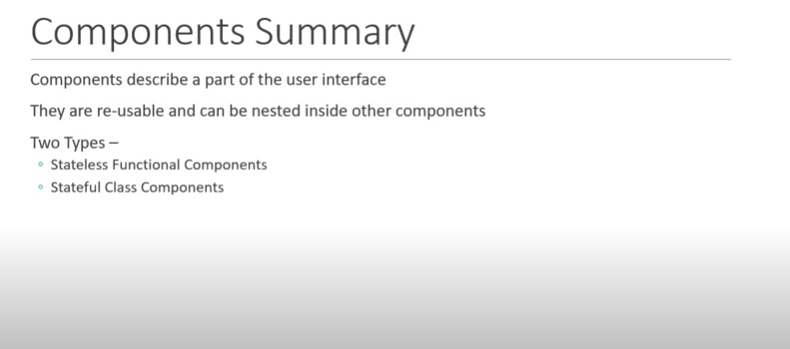


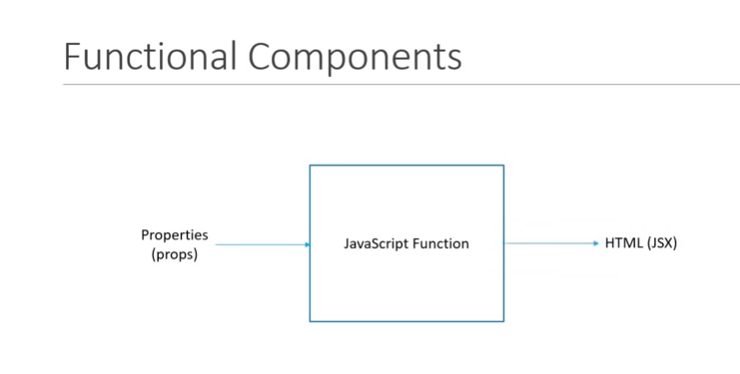


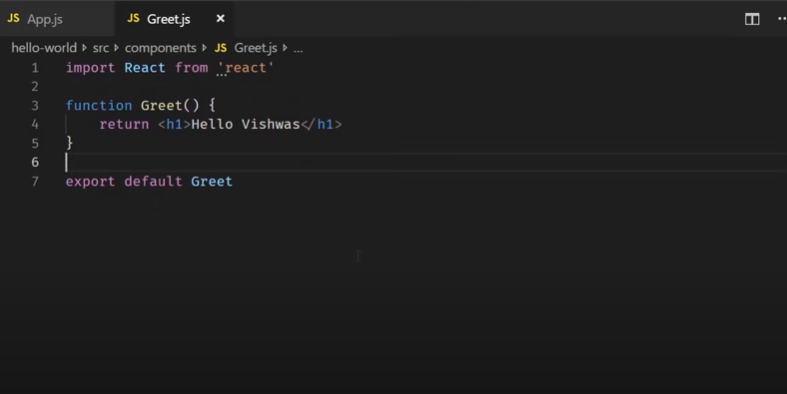


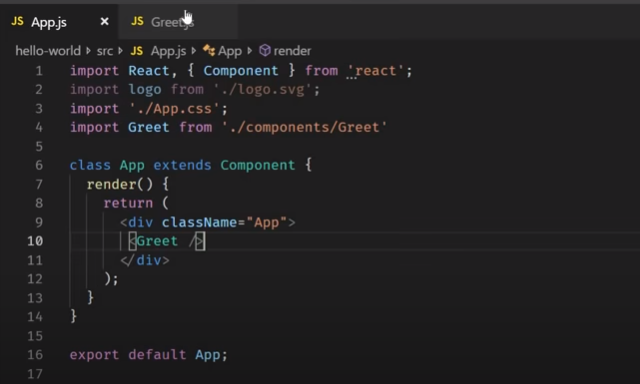


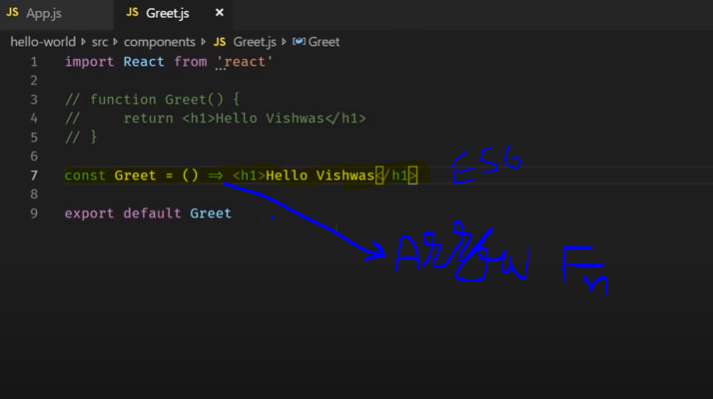
FB have 30,000 Components

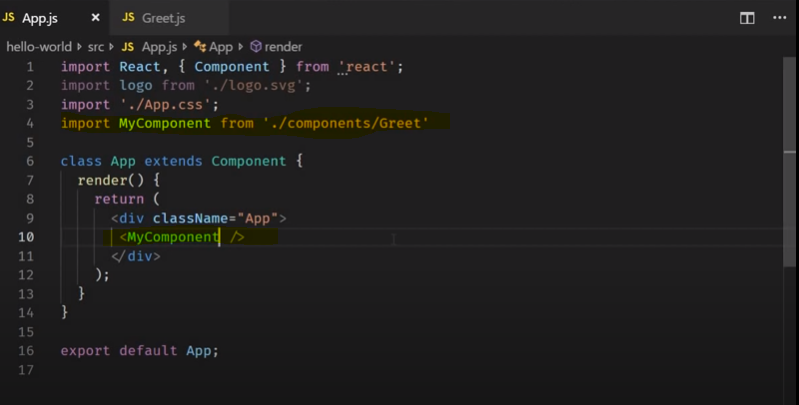


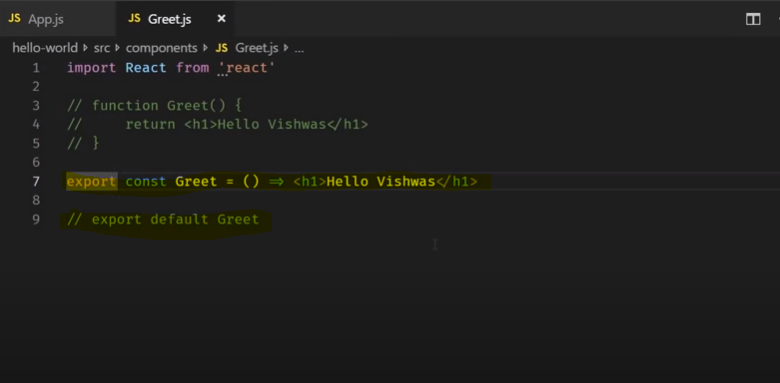


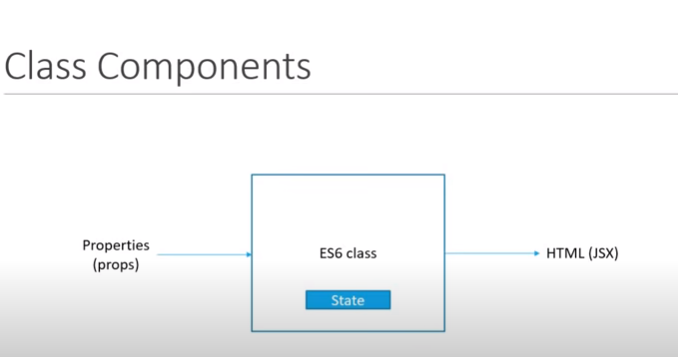


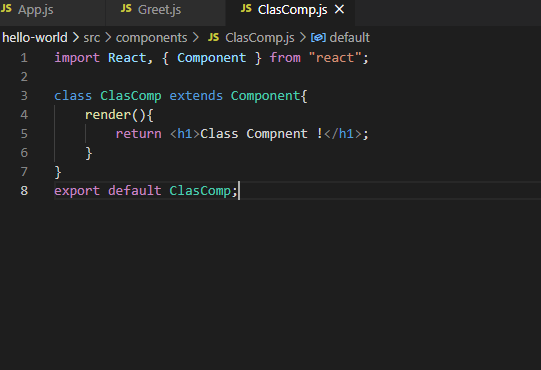


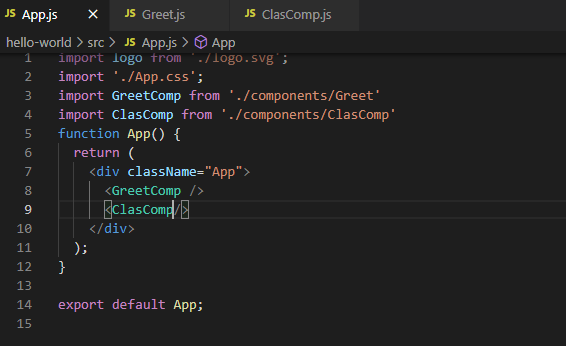


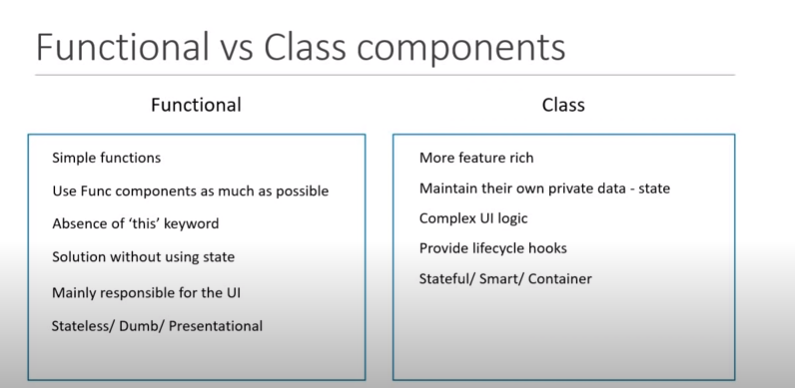


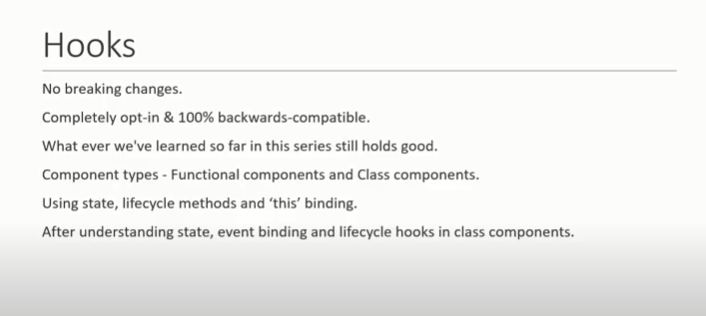


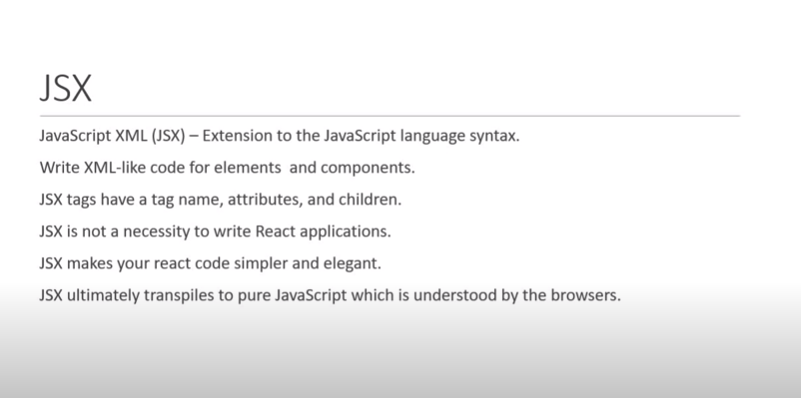


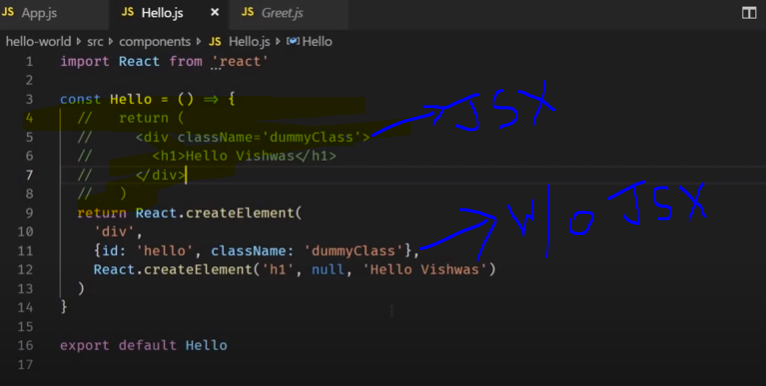














Props :

1. When using props attribute under functional component then write like this.{props.attributeName}
2. When using props attribute under class component then write like this.{this.props.attributeName} and also do not need to pass any parameter in it while accessing in component.

App.js

1. import logo from './logo.svg';
2. import './App.css';
3. import GreetComp from './components/Greet'
4. import ClasComp from './components/ClasComp'
5. function App() {
6. return (
7. <div className="App">
8. <GreetComp name = "Geet" emailId = "geet@gmail.com"/>
9. <GreetComp name = "Suchi"  emailId = "suchi@gmail.com"/>
10. <GreetComp name = "Pallavi"  emailId = "pallavi@gmail.com"/>
11. <ClasComp name = "Pallavi"  emailId = "pallavi@gmail.com"/>
12. <ClasComp name = "Suchi"  emailId = "suchi@gmail.com"/>
13. </div>
14. );
15. }
16. export default App;

Greet.js:

import React from 'react';

const Greet = (props) => {

    console.log(props);

    return (

        <>

        <h1>Function components with simple js and ES6 script !</h1>

        <h2>Reuse same component with passign attibute with value and display</h2>

        <h3>Hello {props.name} \*\*\*\*\*  {props.emailId}</h3>

        </>

    )

}

export default Greet;

ClasComp.js:

import React, { Component } from "react";

class ClasComp extends Component{

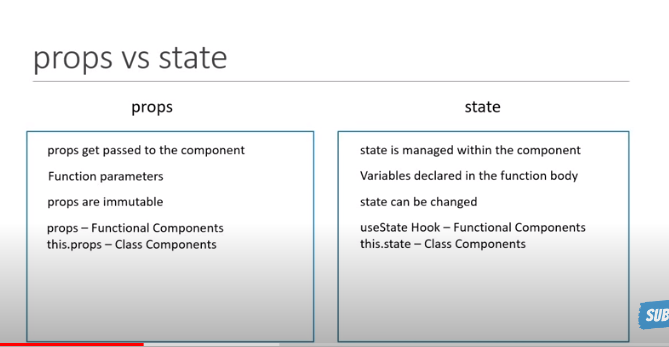
    render(){

        return <h1>Class Compnent {this.props.name} \*\*\*\*\* {this.props.emailId}!</h1>;

    }

}

export default ClasComp;



Message.js

import React, { Component } from "react";

class ClasComp extends Component{

    constructor(){

        super()

        this.state ={

            message:'WelcomeSubciber !'

        }

    }

    changeMsg(){

        this.setState({

            message:'Thank you for subscribe'

        })

    }

    render(){

        return (

            <>

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

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

            </>

        )

    }

}

export default ClasComp;

Include abv component in App.js file then compile it.

SetState : React may grp multiple set state call into single update for better performance as per below e.g.

Counter.js

import React, { Component } from "react";

class Counter extends Component{

    constructor(props){

        super(props)

        this.state = {

            count : 0,

            counterVal:0

        }

    }

    increament(){

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

        this.setState({

            count : this.state.count + 1

        },

        () => {

            console.log('callback function...',this.state.count);//calling callback fn to get exact value

        }

        )

        console.log(this.state.count);//this will start frm 0

    }

    increamentNext(){

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

            counterVal : prevState.counterVal + 1

        }))

        console.log(this.state.counterVal);

    }

    increamentFive(){

        this.increamentNext();

        this.increamentNext();

        this.increamentNext();

        this.increamentNext();

        this.increamentNext();

    }

    render(){

        return (

            <>

                <h1>Count - {this.state.count}</h1>

                <button onClick = {() => this.increament()}>Increament</button>&nbsp;

                <h1>Count Next - {this.state.counterVal}</h1>

                <button onClick = {() => this.increamentFive()}>IncreamentNext</button>

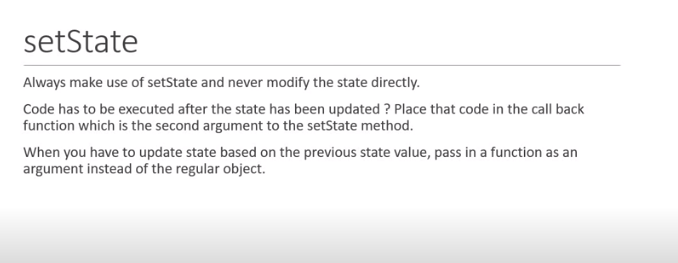
            </>

        )

    }

}

export default Counter;



Destructuring

import React from 'react';

/\* const Greet = (props) => {

    console.log(props);

    return (

        <>

        <h1>Function components with simple js and ES6 script !</h1>

        <h2>Reuse same component with passign attibute with value and display</h2>

        <h3>Hello {props.name} \*\*\*\*\*  {props.emailId}</h3>

        </>

    )

} \*/

//Destructuring our code using props and state

/\* //Below is one way

const Greet = ({name,emailId}) => {

    //console.log(props);

    return (

        <>

        <h1>Function components with simple js and ES6 script !</h1>

        <h2>Reuse same component with assign attibute with value and display</h2>

        <h3>Hello {name} \*\*\*\*\*  {emailId}</h3>

        </>

    )

}   \*/

//Below is second way

const Greet = props => {

    console.log(props);

    const {name,emailId} = props

    return (

        <>

        <h1>Function components with simple js and ES6 script !</h1>

        <h2>Reuse same component with assign attibute with value and display</h2>

        <h3>Hello {name} \*\*\*\*\*  {emailId}</h3>

        </>

    )

}

export default Greet;

import React, { Component } from "react";

// class ClasComp extends Component{

//     render(){

//         return <h1>Class Compnent {this.props.name} \*\*\*\*\* {this.props.emailId}!</h1>;

//     }

// }

//Destructuring our code using props and state

class ClasComp extends Component{

    render(){

        const {name,emailId} = this.props

        // const {state1,state2} = this.state

        return <h1>Class Compnent {name} \*\*\*\*\* {emailId}!</h1>;

    }

}

export default ClasComp;

Exceptional Handling: we can use in two ways functional and class component,please hv a look wid example.

FunctionClick.js

import React from 'react';

function FuntionClick(props) {

    function clickHandler(){

        console.log('Click on function button !');

    }

    return (

        <div>

            <button onClick = {clickHandler}>Click me</button>&nbsp;

            <button onClick = {clickHandler()}>Click me</button>

        </div>

        //second buton : While call handler as function call it will run time when loaded and not work on click

    );

}

export default FuntionClick;

ClassClick.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class ClassClick extends Component {

    constructor(props) {

        super(props);

    }

    ClickHandler(){

        console.log('clicked on class button !');

    }

    render() {

        return (

            <div>

                <button onClick = {this.ClickHandler} >Click Button</button>

            </div>

        );

    }

}

export default ClassClick;

Binding Event Handler: This keyword work in JavaScript.

1. Binding in render
2. Binding with arrow function under render
3. Binding in class constructor
4. Class property as arrow fn.

We cn use abv 3rd option is best approach but we cn use 4th approach as well.

**EventBind.js**

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class EventBind extends Component {

    constructor(props) {

        super(props);

        this.state = {

            message:'Hello'

        }

        //this.clickHandler = this.clickHandler.bind(this);//3rd Approach maily used this way for event bind

    }

    // clickHandler(){

    //     this.setState({

    //         message:'Good bye...'

    //     })

    //     console.log('Good bye...');

    // } //We need to use this fn for 1st and second approach

    clickHandler = () => {

        this.setState({

            message:'Good bye...'

        })

        console.log('Good bye...');

    } //Used this fn for 4th approach

    render() {

        return (

            <div>

                <h3>{this.state.message}</h3>

                {/\* <button onClick = {this.clickHandler.bind(this)}>Click</button> \*/}{/\* 1st Approach \*/}

                {/\* <button onClick = {() =>  this.clickHandler()}>Click</button> \*/}{/\* 2nd Approach \*/}

                {/\* <button onClick = {this.clickHandler}>Click</button> \*/}{/\* 3rd Approach \*/}

                <button onClick = {this.clickHandler}>Click</button>{/\* th Approach \*/}

            </div>

        );

    }

}

export default EventBind;

Method as props : While use props call fn from parent to child and pass value frm child to parent component see below example.

ParentComponent.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

import ChildComponent from './ChildComponent';

class ParentComponent extends Component {

    constructor(props) {

        super(props);

        this.state = {

            parentName : 'Parent'

        }

        this.greetParent = this.greetParent.bind(this);

    }

    greetParent(childName){

      alert(`hello ${this.state.parentName} from ${childName}`);//backticks and dollar curly brackets feature in ES6

    }

    render() {

        return (

            <div>

                <ChildComponent greethandler =  {this.greetParent}/>

            </div>

        );

    }

}

export default ParentComponent;

ChildComponent.js : we are not define state that why use function component rather then class its easy n simple

import React from 'react';

function ChildComponent(props) {

    return (

        <div>

            <button onClick = {() => props.greethandler('Child')}>Greet parent</button>{/\* If we need to send any data frm child to parent then use arrow method while call fn\*/}

        </div>

    );

}

export default ChildComponent;

Conditional Rendering :

UserCondition.js

import React, { Component } from 'react';

class UserConditional extends Component {

    constructor(props) {

        super(props);

        this.state = {

            isLoggedIn : true

        }

    }

    render() {

        // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Below is first approach but not considerable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

        // if(this.state.isLoggedIn === true)

        //     return <div>Welcome Geet !</div>

        //  else

        //     return <div>Welcome Guest !</div> //This is worst approach

        // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Below is second approach but not considerable\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

        // let message;

        // if(this.state.isLoggedIn)

        //     message = <div>Welcome Geet !</div>

        // else

        // message = <div>Welcome Guest !</div>

        // return <div>{message}</div>

        // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Below is Third approach and its considerable using ternary operator\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

        // return this.state.isLoggedIn ? (

        //     <div>Welcome Geet !</div>

        // ) : (

        //     <div> Welcome Guest !</div>

        // )

        // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Below is Forth approach and its considerable using shortest method\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

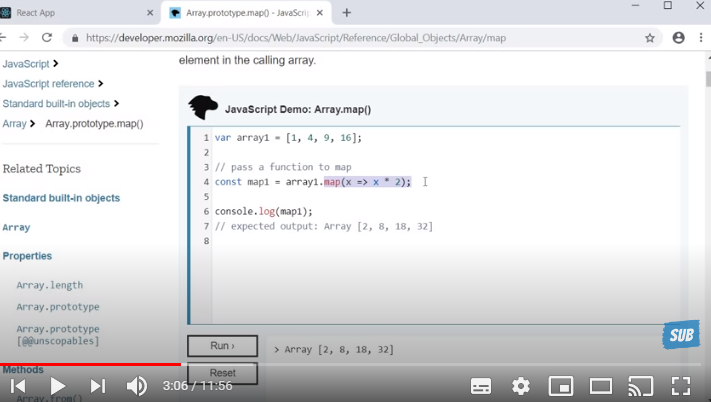
         return  this.state.isLoggedIn && <div>Welcom Geet ! </div> || <div> Welcome Guest !</div>//We don't need to add Or operator if you dn't need to add anything in else condition

    }

}

export default UserConditional;

List rendering :



NameList.js

import React from 'react';

import Person from './Person';

function NameList(props) {

    const persons = [

        {

            id:1,

            name:'Geet',

            age:28,

            skill:'ReactJs'

        },

        {

            id:2,

            name:'Pallavi',

            age:25,

            skill:'Angular Js'

        },

        {

            id:3,

            name:'Suchitra',

            age:25,

            skill:'Java'

        },

    ]

    const personList = persons.map(person => <Person person = {person} />)

    return (

        <div>

            {personList}

        </div>

    );

}

export default NameList;

Person.js

import React from 'react';

function Person({person}) {

    return (

        <div>

            <h2>I am {person.name}. I am {person.age} year old.I know {person.skill}</h2>

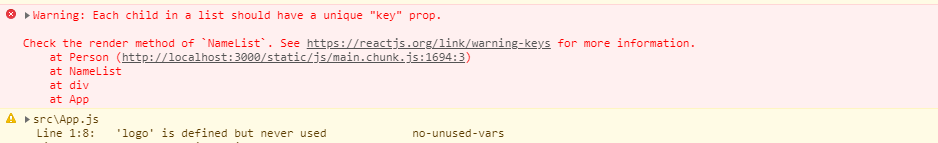
        </div>

    );

}

export default Person;

Lists and Keys In react:



import React from 'react';

import Person from './Person';

function NameList(props) {

    const persons = [

        {

            id:1,

            name:'Geet',

            age:28,

            skill:'ReactJs'

        },

        {

            id:2,

            name:'Pallavi',

            age:25,

            skill:'Angular Js'

        },

        {

            id:3,

            name:'Suchitra',

            age:25,

            skill:'Java'

        },

    ]

    const personList = persons.map(person => <Person key = {person.id} person = {person} />)//Keuy always be unique doest matter it is id or name but always it should be unique and if we not pass key prop then it will through warning unique key prop.

    return (

        <div>

            {personList}

        </div>

    );

}

export default NameList;

Key props is not accessible in child component.

Person.js

import React from 'react';

function Person({person, key}) {

    return (

        <div>

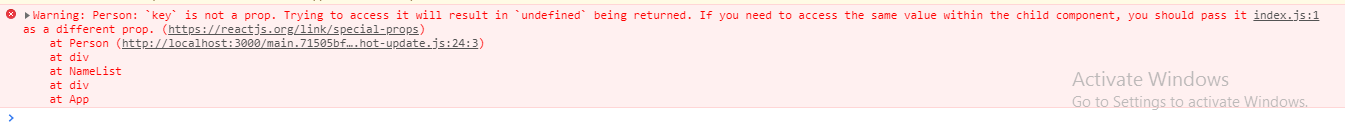
            <h2>{key}I am {person.name}. I am {person.age} year old.I know {person.skill}</h2>{/\*chk on ocnsole itwill show warning msg we cant pass key as a prop in childcomponent, we should need to pass in diff prop name \*/}

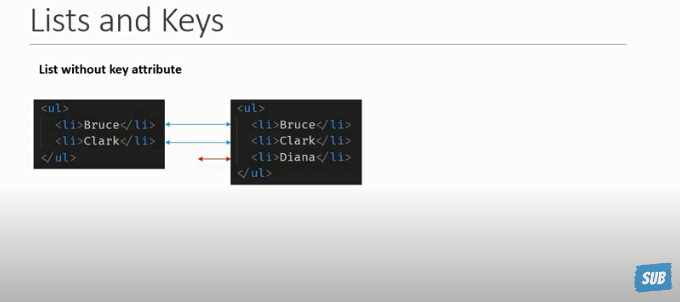
        </div>

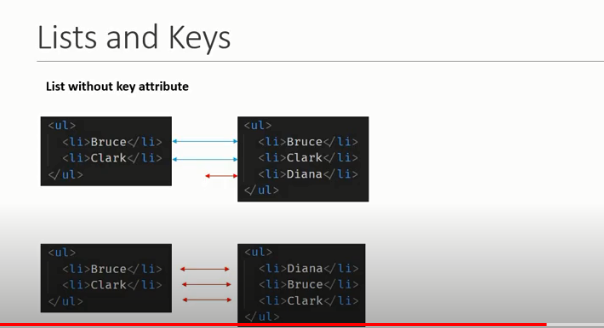
    );

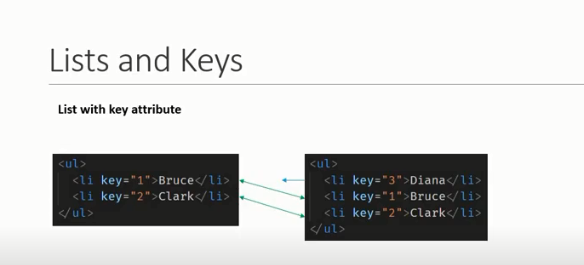
}

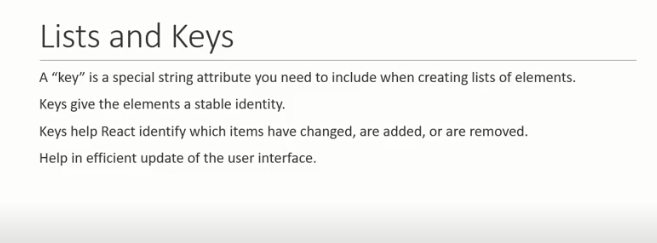
export default Person;











nameList.js : Using index using normal array

import React from 'react';

import Person from './Person';

function NameList(props) {

const names = ['Geet','Pallavi','Suchi']

const namelist = names.map((name,index) => <h2 key = {index}>{name}</h2>)

    // const persons = [

    //     {

    //         id:1,

    //         name:'Geet',

    //         age:28,

    //         skill:'ReactJs'

    //     },

    //     {

    //         id:2,

    //         name:'Pallavi',

    //         age:25,

    //         skill:'Angular Js'

    //     },

    //     {

    //         id:3,

    //         name:'Suchitra',

    //         age:25,

    //         skill:'Java'

    //     },

    // ]

    // const personList = persons.map(person => <Person key = {person.id} person = {person} />)//Keuy always be unique doest matter it is id or name but always it should be unique and if we not pass key prop then it will through warning unique key prop.

    return (

        <div>

            {namelist}

        </div>

        // <div>

        //     {personList}

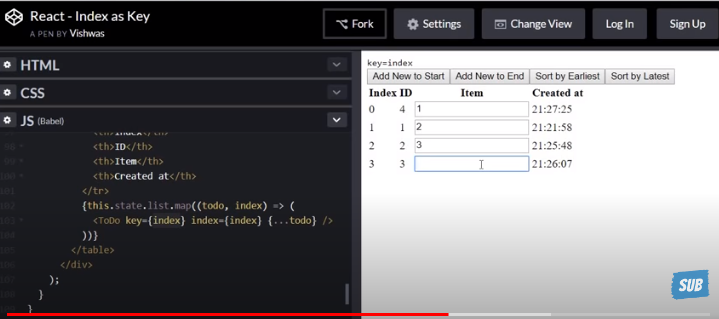
        // </div>

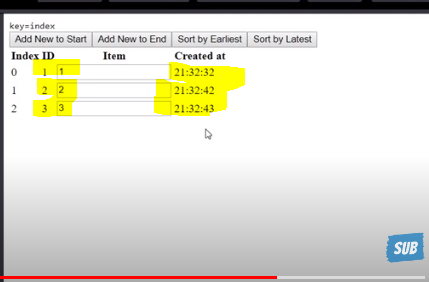
    );

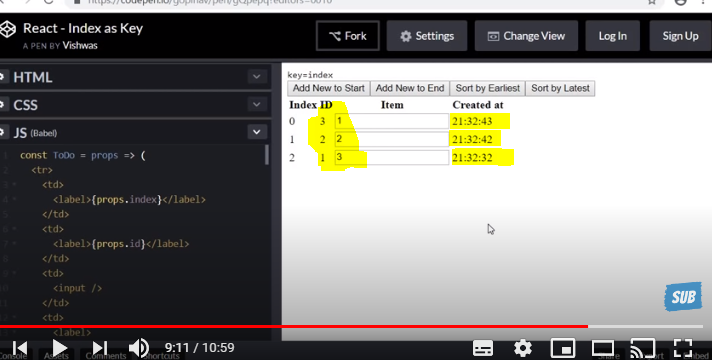
}

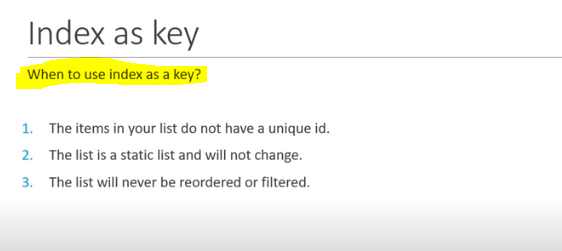
export default NameList;

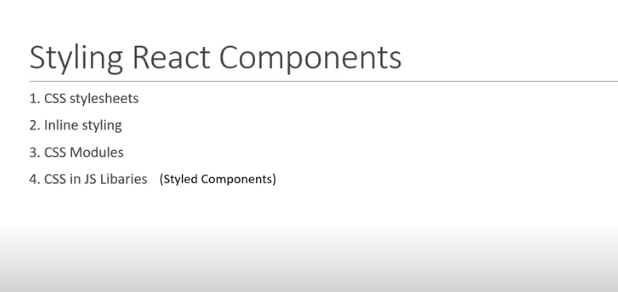
As per below example facing issue if value is blank











Stylesheet.js

import React from 'react';

import './mystyle.css'

function Stylesheet(props) {

    let className = props.primary ? 'primary' : ''

    return (

        <div>

            <h4  className = {`${className} font-xl`}>Stylesheet</h4>{/\*/using backstick we cn pass multiple class  wid combination of dynamic and static class\*/}

        </div>

    );

}

export default Stylesheet;

mystyle.css

.primary{

    color:orange;

}

.font-xl{

    font-size: xx-large;

}

Inline css :

Inline.js

import React from 'react';

function Inline(props) {

    const heading = {

        fontSize:'72px',//always write css attribute in camelcase while suing inline css with passing as object

        color:'blue'

    }

    return (

        <div>

            <h4 style = {heading}>Inline styling</h4>

        </div>

    );

}

export default Inline;

Css Module : Classes are locally scoped by default;

Ex if we hv created any css file and include in app.js file then their classes we cn use in any child components. If you r using class from css module file so that class you can’t be used in child component.

import React from 'react';

function Inline(props) {

    const heading = {

        fontSize:'72px',//always write css attribute in camelcase while suing inline css with passing as object

        color:'blue'

    }

    return (

        <div>

            <h2 className = 'error'>Error from inline component</h2>

            <h4 style = {heading}>Inline styling</h4>

        </div>

    );

}

export default Inline;

import logo from './logo.svg';

import './App.css';

import GreetComp from './components/Greet'

import ClasComp from './components/ClasComp'

import Message from './components/Message'

import Counter from './components/Counter'

import FuntionClick from './components/FuntionClick';

import ClassClick from './components/ClassClick';

import EventBind from './components/EventBind';

import ParentComponent from './components/ParentComponent';

import UserConditional from './components/UserConditional';

import NameList from './components/NameList';

import Stylesheet from './components/Stylesheet';

import Inline from './components/Inline';

import './appStyles.css'

import styles from './appStyles.module.css'

function App() {

  return (

    <div className="App">

      <h2 className = 'error'>Error</h2>

      <h2 className = {styles.success}>Success</h2>

      <Inline/>

      {/\* <Stylesheet primary = {true}/> \*/}

      {/\* <NameList/> \*/}

      {/\* <UserConditional /> \*/}

      {/\* <ParentComponent /> \*/}

      {/\* <EventBind/> \*/}

      {/\* <FuntionClick />

      <ClassClick />

      <GreetComp name = "Geet" emailId = "geet@gmail.com"/>

      <GreetComp name = "Suchi"  emailId = "suchi@gmail.com"/>

      <GreetComp name = "Pallavi"  emailId = "pallavi@gmail.com"/>

      <ClasComp name = "Pallavi"  emailId = "pallavi@gmail.com"/>

      <ClasComp name = "Suchi"  emailId = "suchi@gmail.com"/>

      <Message />

      <Counter/><br/><br/><br/>\*/}

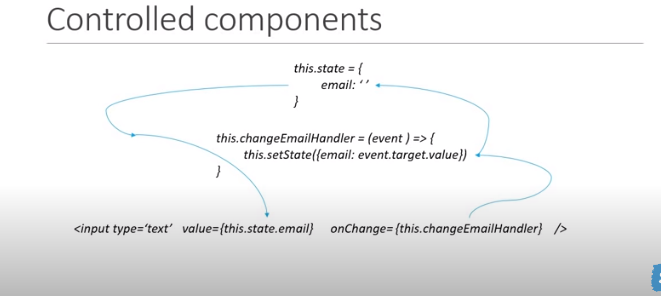
    </div>

  );

}

export default App;

Basics of form elements:

Form.js

import React, { Component } from 'react';

class Form extends Component {

    constructor(props) {

        super(props);

        this.state = {

            username:'',

            comment:'',

            topic:''

        }

    }

    handleUsername = event =>{

        this.setState({

            username:event.target.value

        })

    }

    handleComment = event =>{

        this.setState({

            comment:event.target.value

        })

    }

    handleTopic = event => {

        this.setState({

            topic:event.target.value

        })

    }

    handleSubmit = event => {

        alert(`${this.state.username} ${this.state.comment} ${this.state.topic}`);

        event.preventDefault();

    }

    render() {

        return (

           <form onSubmit = {this.handleSubmit}>

                <div>

                    <label>Username</label>

                    <input type='text' value = {this.state.username} onChange = {this.handleUsername}/>

                </div>

                <br/>

                <div>

                    <label>Comment:</label>

                    <textarea value = {this.state.comment} onChange = {this.handleComment}/>

                </div>

                <br/>

                <div>

                    <label>Topic:</label>

                    <select value = {this.state.topic} onChange = {this.handleTopic}>

                        <option value = "react">React</option>

                        <option value = "angular">Angular</option>

                        <option value = "typescript">Typescript</option>

                    </select>

                </div>

                <br/>

                <button type = "submit">Submit</button>

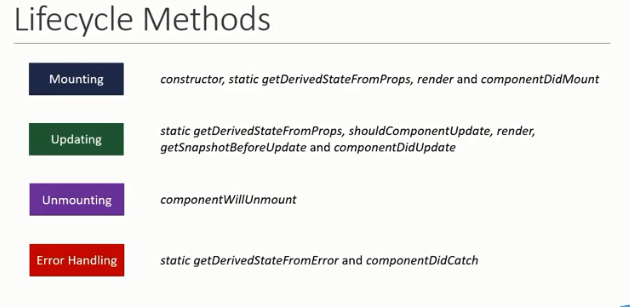
            </form>

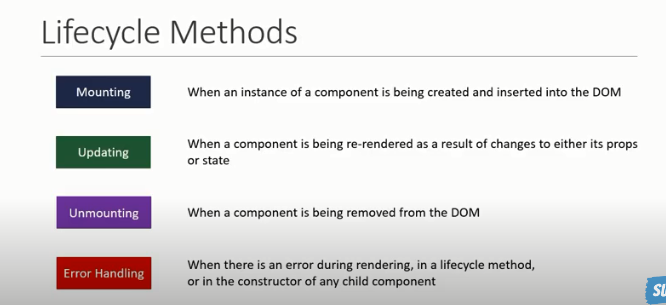
        );

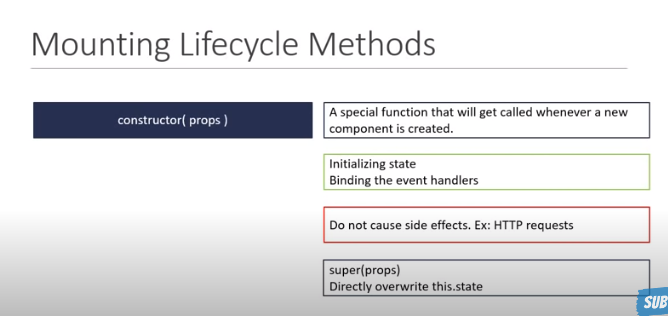
    }

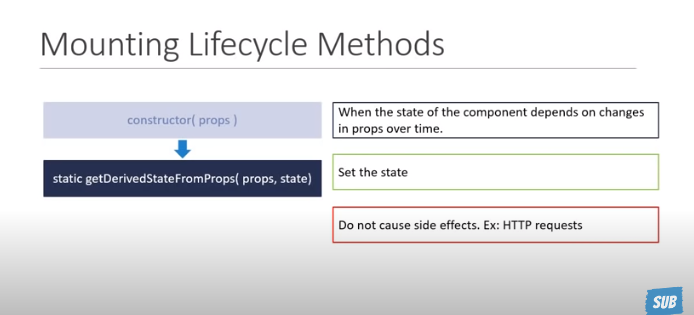
}

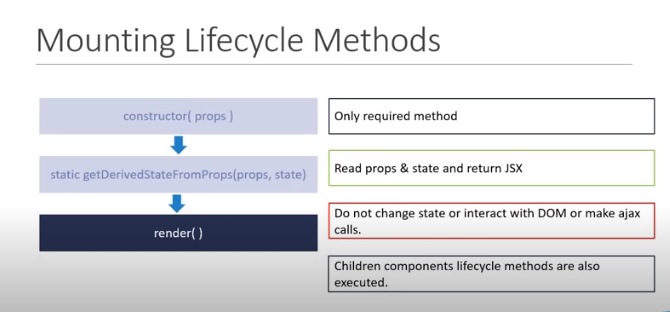
export default Form;

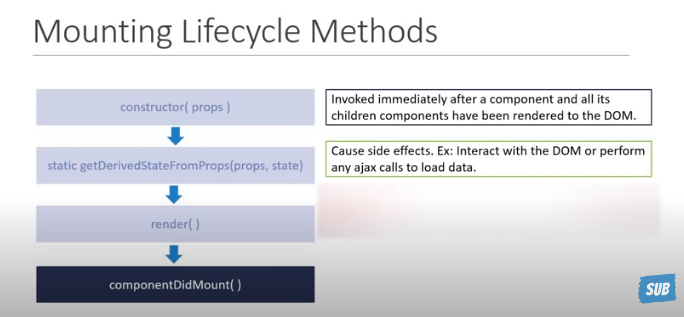












LifecycleA.js

import React, { Component } from 'react';

import LifeCycleB from './LifeCycleB';

class LifeCycleA extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

        console.log('LifecycleA Constructor');

    }

    componentDidMount() {

        console.log('LifecycleA ComponentDidMount');

    }

    static getDerivedStateFromProps(props,state){

        console.log('LifecycleA getDerivedStateFromProps');

        return null;

    }

    render() {

        console.log('LifecycleA Render');

        return (

            <>

            <div>

                LifeCycleA

            </div>

            <LifeCycleB/>

            </>

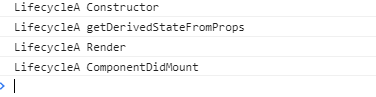
        );

    }

}

export default LifeCycleA;

output:



LifeCycleB.js : include tis component to LifeCycleB..js

import React, { Component } from 'react';

class LifeCycleB extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

        console.log('LifeCycleB Constructor');

    }

    componentDidMount() {

        console.log('LifeCycleB ComponentDidMount');

    }

    static getDerivedStateFromProps(props,state){

        console.log('LifeCycleB getDerivedStateFromProps');

        return null;

    }

    render() {

        console.log('LifeCycleB Render');

        return (

            <div>

                LifeCycleB

            </div>

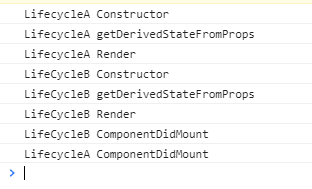
        );

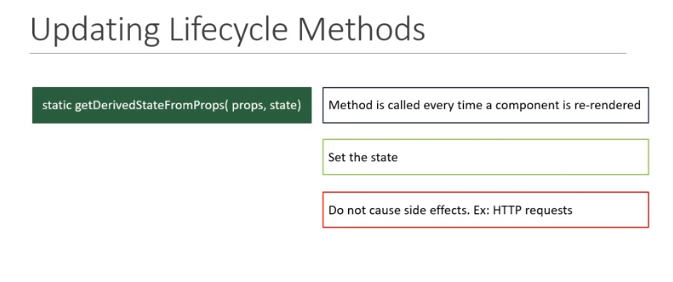
    }

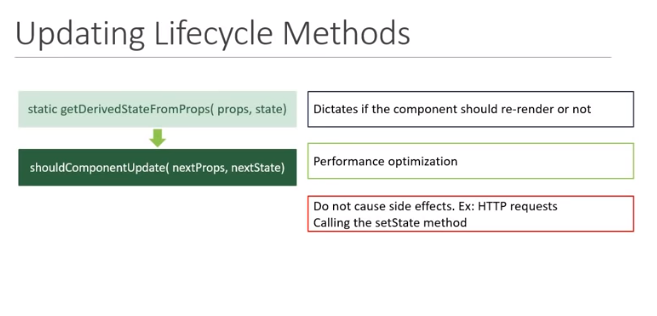
}

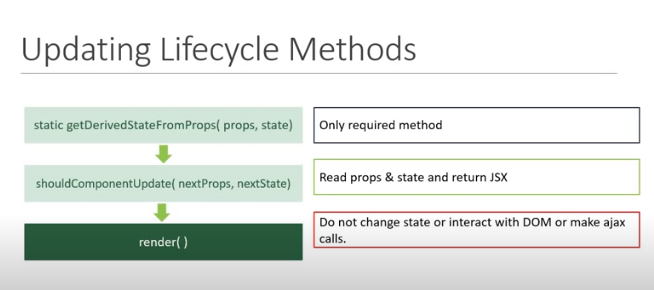
export default LifeCycleB;

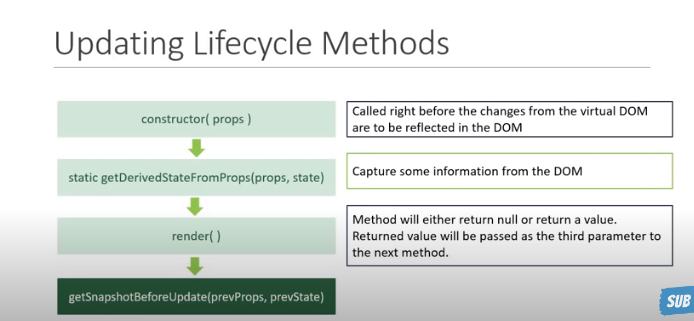
Output:

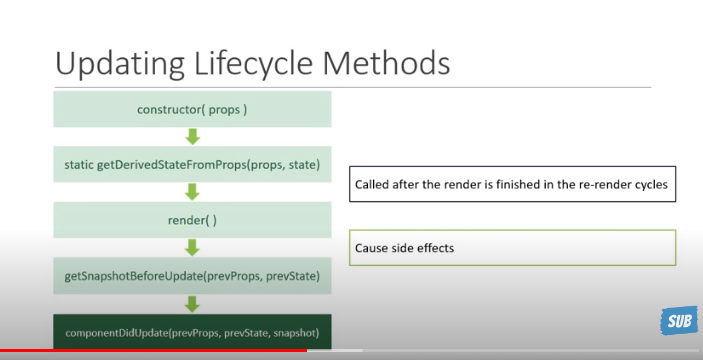












Before calling update fn result is :

import React, { Component } from 'react';

import LifeCycleB from './LifeCycleB';

class LifeCycleA extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

        console.log('LifecycleA Constructor');

    }

    componentDidMount() {

        console.log('LifecycleA ComponentDidMount');

    }

    static getDerivedStateFromProps(props,state){

        console.log('LifecycleA getDerivedStateFromProps');

        return null;

    }

    shouldComponentUpdate(){

        console.log('LifecycleA shouldComponentUpdate');

        return true;

    }

    getSnapshotBeforeUpdate(){

        console.log('LifecycleA getSnapshotBeforeUpdate');

        return null;

    }

    componentDidUpdate(){

        console.log('LifecycleA componentDidUpdate');

    }

    changeState = () =>{

        this.setState({

            name:'Codevolution'

        })

    }

    render() {

        console.log('LifecycleA Render');

        return (

            <>

            <div>

                LifeCycleA

            </div>

            <button onClick = {this.changeState}>Change State</button>

            <LifeCycleB/>

            </>

        );

    }

}

export default LifeCycleA;

import React, { Component } from 'react';

class LifeCycleB extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

        console.log('LifeCycleB Constructor');

    }

    componentDidMount() {

        console.log('LifeCycleB ComponentDidMount');

    }

    static getDerivedStateFromProps(props,state){

        console.log('LifeCycleB getDerivedStateFromProps');

        return null;

    }

    shouldComponentUpdate(){

        console.log('LifeCycleB shouldComponentUpdate');

        return true;

    }

    getSnapshotBeforeUpdate(){

        console.log('LifeCycleB getSnapshotBeforeUpdate');

        return null;

    }

    componentDidUpdate(){

        console.log('LifeCycleB componentDidUpdate');

    }

    render() {

        console.log('LifeCycleB Render');

        return (

            <div>

                LifeCycleB

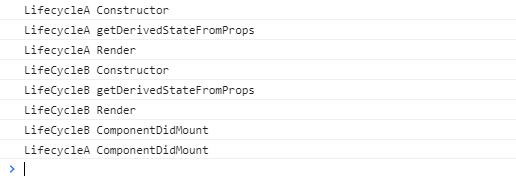
            </div>

        );

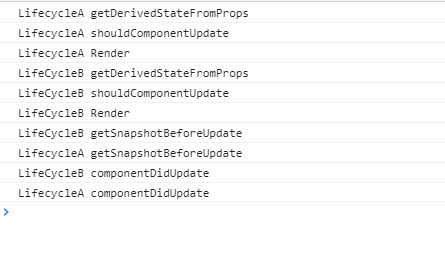
    }

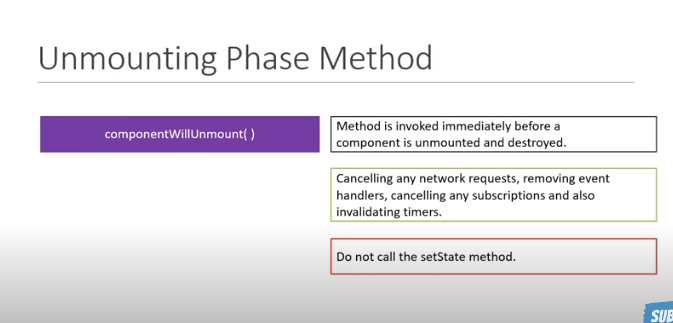
}

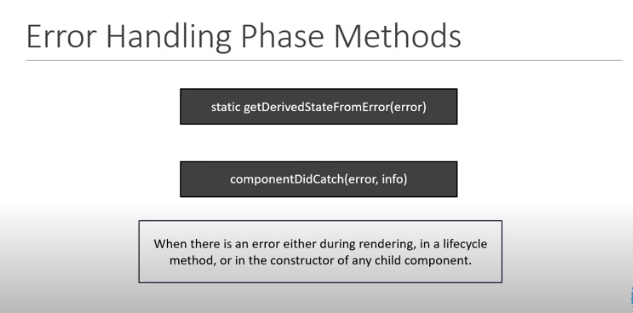
export default LifeCycleB;



After click on button and result below:







Fragments: Group list of children element without adding extra node to the DOM.

e.g : <></>

<React.Fragment key = {item.id}></React.Fragment> : While using this u cn add key in react fragment if needed otherwise we cn use blank bracket like as abv.

Pure Components: This is not re-render

ParentComp.js

import React, { Component } from 'react';

import PureComp from './PureComp';

import RegComp from './RegComp';

class ParentComp extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

    }

    componentDidMount() {

        setInterval(() => {

            this.setState({

                name:'Geet'

            })

        },2000);

    }

    render() {

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Component\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

        return (

            <div>

                <h2>Parent Component</h2>

               <RegComp name = {this.state.name}/>

               <PureComp name= {this.state.name}/>

            </div>

        );

    }

}

export default ParentComp;

RegComp.js

import React, { Component } from 'react';

class RegComp extends Component {

    render() {

        console.log('Regular Component');

        return (

            <div>

                Regular Component {this.props.name}

            </div>

        );

    }

}

export default RegComp;

PureComp.js

import React, { PureComponent } from 'react';

class PureComp extends PureComponent {

    render() {

        console.log('Pure Component');

        return (

            <div>

                Pure Components {this.props.name}

            </div>

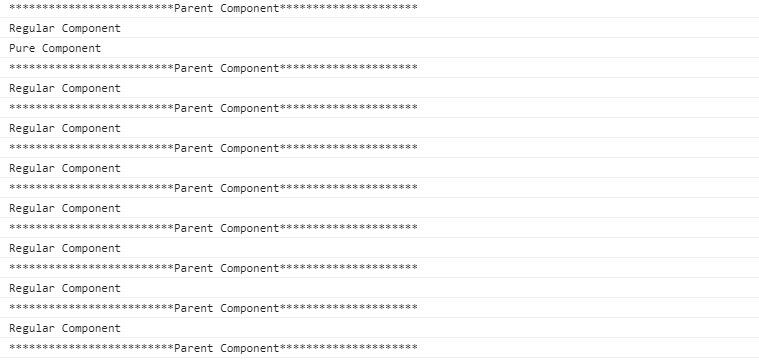
        );

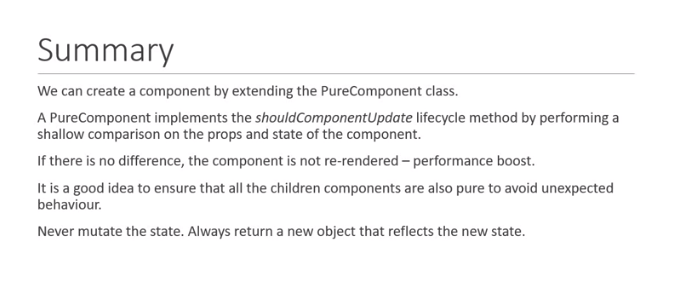
    }

}

export default PureComp;

Output: here only regular component will re-render not pure component.





import React from 'react';

function MemoComp({name}) {

    return (

        <div>

            Memo Component

            {name} !

        </div>

    );

}

export default React.memo(MemoComp);

import React, { Component } from 'react';

import MemoComp from './MemoComp';

import PureComp from './PureComp';

import RegComp from './RegComp';

class ParentComp extends Component {

    constructor(props) {

        super(props);

        this.state = {

            name:'Geet'

        }

    }

    componentDidMount() {

        setInterval(() => {

            this.setState({

                name:'Geet'

            })

        },2000);

    }

    render() {

        console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parent Component\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

        return (

            <div>

                <h2>Parent Component</h2>

                <MemoComp name = {this.state.name}/>

               {/\* <RegComp name = {this.state.name}/>

               <PureComp name= {this.state.name}/> \*/}

            </div>

        );

    }

}

export default ParentComp;

React memo also not re-render and its working in function component!

RefDemo.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

class RefDemo extends Component {

    constructor(props) {

        super(props);

        this.inputRef = React.createRef();

    }

    componentDidMount() {

        this.inputRef.current.focus();//when we willopen form and focus at least one field i.e. need focus cursor

        //console.log(this.inputRef);

    }

    clickhandler = () => {

        alert(this.inputRef.current.value);

    }

    render() {

        return (

            <div>

                <input type="text" ref = {this.inputRef}/>&nbsp;

                <button onClick = {this.clickhandler}>Click</button>

            </div>

        );

    }

}

export default RefDemo;

Input.js

import React, { Component } from 'react';

class Input extends Component {

    constructor(props) {

        super(props);

        this.inputRef = React.createRef();

    }

    focusInput() {

        this.inputRef.current.focus();

    }

    render() {

        return (

            <div>

                <input type="text" ref = {this.inputRef}/>

            </div>

        );

    }

}

export default Input;

FocusInput.js

import React, { Component } from 'react';

import PropTypes from 'prop-types';

import Input from './Input';

class FocusInput extends Component {

    constructor(props) {

        super(props);

        this.componentRef = React.createRef();

    }

    clickHandler = () => {

        this.componentRef.current.focusInput();

    }

    render() {

        return (

            <div>

                <Input ref = {this.componentRef}/>

                <button onClick = {this.clickHandler}>Focus Input</button>

            </div>

        );

    }

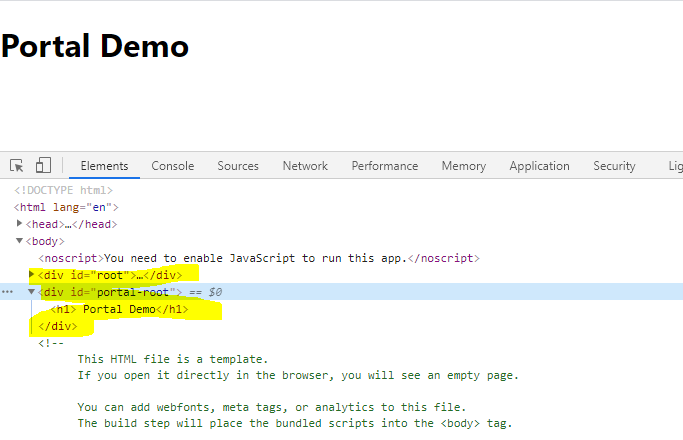
}

FocusInput.propTypes = {

};

export default FocusInput;

Portals: Ability to provide break DOM Tree.



<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <div id="portal-root"></div>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>

PortalDemo.js

import React from 'react';

import ReactDOM from 'react-dom'

function PortalDemo(props) {

    return ReactDOM.createPortal (

        <h1> Portal Demo</h1>,

        document.getElementById('portal-root')

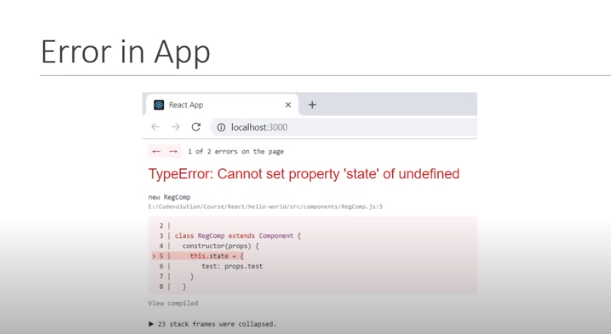
    );

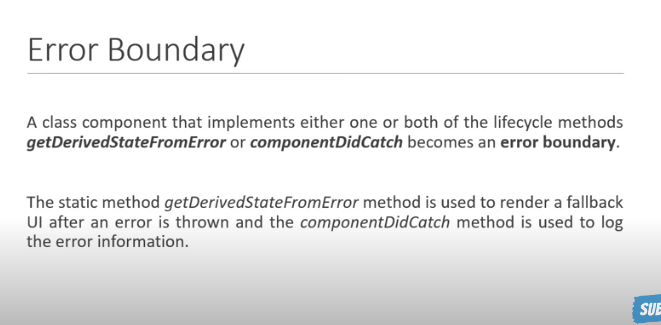
}

export default PortalDemo;

Error Boundaries :

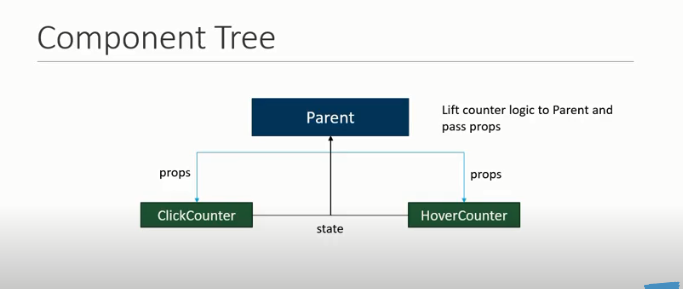


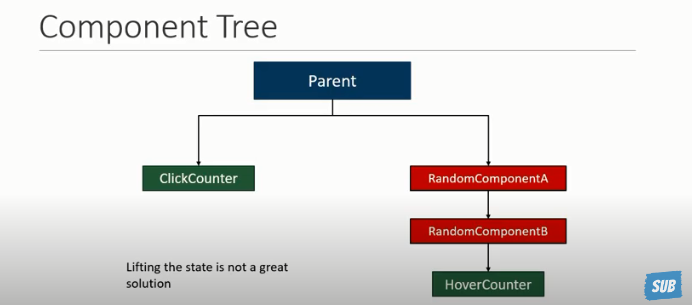






Higher Order Component Part 1:





How to reuse our code rather writing same logic in diff components it’s called higher order component.

ClickCounter.js

import React, { Component } from 'react';

class ClickCounter extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0

        }

    }

    increamentCounter = () => {

        this.setState(prevState => {

            return {count:prevState.count + 1}

        })

    }

    render() {

        const {count} = this.state;

        return (

            <div>

                <button onClick = {this.increamentCounter}>Click Count {count}</button>

            </div>

        );

    }

}

export default ClickCounter;

HoverCounter.js

import React, { Component } from 'react';

class HoverCounter extends Component {

    constructor(props) {

        super(props);

        this.state = {

            count:0

        }

    }

    increamentCounter = () => {

        this.setState(prevState => {

            return {count:prevState.count + 1}

        })

    }

    render() {

        const {count} = this.state

        return (

            <h2 onMouseOver = {this.increamentCounter}>

                Counter {count}

            </h2>

        );

    }

}

export default HoverCounter;

**WithCounter.js**: Using this component we are using HOC(Higher ordered component ) process for creating common counter function and reuse in all diff component where we need.

import React, { Component } from 'react';

const UpdatedComponent = OrignalComponent => {

    class NewComponent extends React.Component{

        constructor(props) {

            super(props);

            this.state = {

                count:0

            }

        }

        increamentCounter = () => {

            this.setState(prevState => {

                return {count:prevState.count + 1}

            })

        }

        render (){

            const {count} = this.state

            return <OrignalComponent count = {count} increaCount = {this.increamentCounter}/>

        }

    }

    return NewComponent

}

export default UpdatedComponent

ClickCounter.js

import React, { Component } from 'react';

import UpdatedComponent from './WithCounter'

class ClickCounter extends Component {

    // constructor(props) {

    //     super(props);

    //     this.state = {

    //         count:0

    //     }

    // }

    // increamentCounter = () => {

    //     this.setState(prevState => {

    //         return {count:prevState.count + 1}

    //     })

    // }

    render() {

        //const {count} = this.state;

        const {count,increaCount} = this.props;

        return (

            <div>

                <button onClick = {increaCount}>  Click Count {count}</button>

            </div>

        );

    }

}

export default UpdatedComponent(ClickCounter);

HoverCounter.js

import React, { Component } from 'react';

import UpdatedComponent from './WithCounter'

class HoverCounter extends Component {

    render() {

        const {count,increaCount} = this.props;

        return (

            <h2 onMouseOver = {increaCount} style = {{cursor:'pointer',backgroundColor : 'royalblue'}}>

                Counter {count}

            </h2>

        );

    }

}

export default UpdatedComponent(HoverCounter);

As per HOC we need to keep same component name as well as same class Component name.

WithCounter.js

import React, { Component } from 'react';

const withCounter = WrappedComponent => {//naming convection

    class WithCounter extends React.Component{

        constructor(props) {

            super(props);

            this.state = {

                count:0

            }

        }

        increamentCounter = () => {

            this.setState(prevState => {

                return {count:prevState.count + 1}

            })

        }

        render (){

            const {count} = this.state

            return <WrappedComponent count = {count} increaCount = {this.increamentCounter}/>

        }

    }

    return WithCounter

}

export default withCounter

When we will pass any props from ClickCounter.js component using HOC so we hv to pass props from HOC component we can’t directly use props from respective component because we are following HOC process.

See below e.g.

App.js

<ClickCounter name = 'Geet'/>

ClickCounter.js

  <button onClick = {increaCount}>{this.props.name}  Click Count {count}</button>

WithCounter.js

 return <WrappedComponent count = {count} increaCount = {this.increamentCounter} {...this.props}/>

If we need to pass parameter using HOC pls see below e.g.

WithCounter.js

const withCounter = (WrappedComponent,increamentNum )=> {//naming convection

return {count:prevState.count + increamentNum}

ClickCounter.js

export default withCounter(ClickCounter,10);

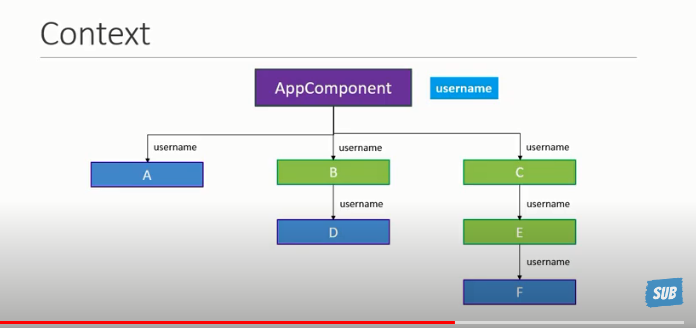
HoverCounter.js

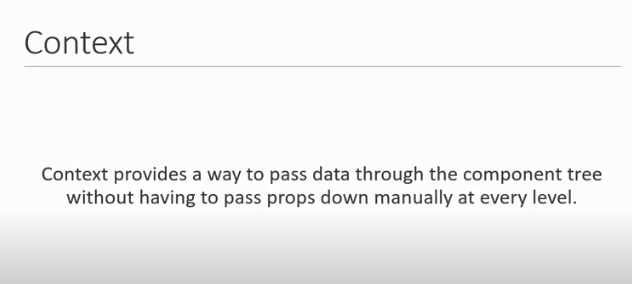
export default withCounter(HoverCounter,5);

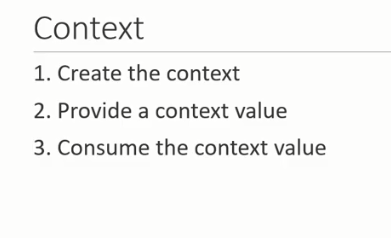
**Render props:** We cn create common functionality using render props as well pls see below e.g.



Context api :







App.js

<UserProvider value = 'Geet'>

          <ComponentC />

      </UserProvider>

ComponentC.js

import React, { Component } from 'react';

import ComponentE from './ComponentE';

class ComponentC extends Component {

    constructor(props) {

        super(props);

    }

    render() {

        return (

            <ComponentE/>

        );

    }

}

ComponentC.propTypes = {

};

export default ComponentC;

ComponentE.js

import React, { Component } from 'react';

import ComponentF from './ComponentF';

import { UserContext } from './UserContext';

class ComponentE extends Component {

    constructor(props) {

        super(props);

    }

    static contextType= UserContext

    render() {

        return (

            <div>

                Componet E {this.context}

                <ComponentF/>

            </div>

        );

    }

}

export default ComponentE;

ComponentF.js

import React, { Component } from 'react';

import { UserConsumer } from './UserContext';

class ComponentF extends Component {

    constructor(props) {

        super(props);

    }

    render() {

        return (

            <UserConsumer>

                {name => {

                    return <div>Hello {name}</div>

                }}

            </UserConsumer>

        );

    }

}

export default ComponentF;

UserContext.js

import React from 'react'

const UserContext = React.createContext();

const UserProvider = UserContext.Provider;

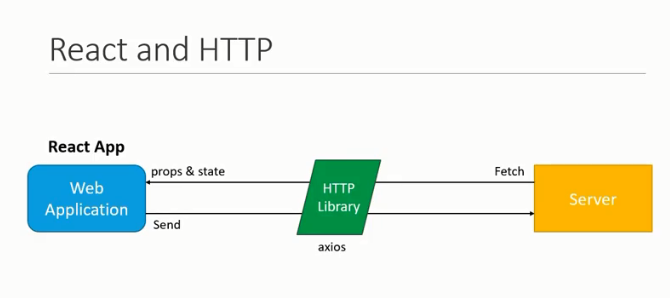
const UserConsumer = UserContext.Consumer;

export {UserContext,UserProvider,UserConsumer}



React and HTTP :





Run Command for using Https request : npm install axios

JSON Placeholder: Fake online rest API for testing and prototyping

<https://jsonplaceholder.typicode.com/>

GetMethod.js

import React, { Component } from 'react';

import axios from 'axios'

class GetMethod extends Component {

    constructor(props) {

        super(props);

        this.state = {

            posts:[],

            errormsg:''

        }

    }

    componentDidMount() {

        axios.get('https://jsonplaceholder.typicode.com/posts')

        .then(response => {

            console.log(response.data);

            this.setState({

                posts:response.data

            })

        })

        .catch(error => {

            console.log(error);

            this.setState({

                errormsg:'Error retrieving data'

            })

        })

    }

    render() {

        const {posts,error} = this.state

        return (

            <>

                <div>

                   <h2> List of Posts</h2><hr/>

                    <ol>

                    {posts.length ?

                        posts.map(post => <li style = {{textAlign:'left'}} key = {post.id}>{post.title}</li>)

                    : null}

                    </ol>

                </div>

                {error ? <div>{error}</div> : null}

            </>

        );

    }

}

export default GetMethod;

import React, { Component } from 'react';

import axios from 'axios'

class PostMethod extends Component {

    constructor(props) {

        super(props);

        this.state = {

            userId: '',

            title: '',

            body: ''

        }

    }

    changeHandler = e => {

        this.setState({

            [e.target.name]:e.target.value

        })

    }

    submitHandler = e => {

        e.preventDefault();

        console.log(this.state);

        axios.post('https://jsonplaceholder.typicode.com/posts',this.state)

        .then(response => {

            console.log(response);

        })

        .catch(error => {

            console.log(error);

        })

    }

    render() {

        const {userId,title,body} = this.state

        return (

            <div>

                <form onSubmit = {this.submitHandler}>

                    <div>

                        <label>UserId</label>

                        <input

                        type = "text"

                        name = "userId"

                        value = {userId }

                        onChange = {this.changeHandler}/>

                    </div>

                    <div>

                        <label>Title</label>

                        <input

                        type = "text"

                        name = "title"

                        value = {title}

                        onChange = {this.changeHandler}/>

                    </div>

                    <div>

                        <label>Body</label>

                        <input

                        type = "text"

                        name = "body"

                        value = {body}

                        onChange = {this.changeHandler}/>

                    </div>

                    <button type="submit">Submit</button>

                </form>

            </div>

        );

    }

}

export default PostMethod;

