React Notes and Definition

❖ Component

- Component UI ko alag alag pieces me devide kar ne ki functionaly deta hai. or ush ka jo code hai isolated hai.dushare componanet ko effect nahi karta.

❖ Router

- ek component se dushre component me link kar ne ke liye Router use hota hai.

❖ Render

- Render always return JSX.

❖ lazy-loading

- jab hum Application ko open karte hai tab sare data ek sath load hote hai. tab application ke open honeke time pe hame koi data prin karvana ho or ek sath sabhi data ko load karvye bina one by one data ko load karvana ho tabhi lazy loading use hota hai.

❖ Class component

- Class component containg a render method which helps to return jsx is called class component. Class component ko extend karna padta hai React.component. Class component ke inside me render method hoti hai jo JSX ko return karti hai.

❖ JSX

- JSX stands for JavaScript Syntax Extension JSX is used to combine JavaScript and HTML both.

❖ State

- State is a React JS variable. It is mutable it means when we want to change the data into runtime then we use state.

❖ Props

- Jab hame koi data ko pass karna hai from parent component to child component then we use props.

❖ State Life Cycle

- It is a way to work like how class components are working after and before the class component are render there are three phrases Mounting Updating and Unmounting

- In Mounting phrase whenever our component is created and load at first time then mounting is happen is just like render.

- In Updating phrase when our state and props are update at that time updating methods are called.
- In Unmounting whenever our we move one component to another component or destroy the component then unmount method are used.

ComponentWillMount

Jab hum koi component ko create karte hai render hone se pehle jab koi data ko print karvana ho tub yai method use hoti hai it is same like constructor.

componentDidMount

When the component is created after these method load.

The componentDidMount() method is called after the component is rendered.

componentWillReceiveProps:-Jab hum koi component ko data bhejte hai tab props dravra bhejte hai to ush ke pehle hi props receive hota hai ya nahi wo check karvata ho tab ye method use hoti hai.

shouldComponentUpdate- Jan hum state ya props ko update karvae hai tab yai method true and false ko return karta hai by default yai method true return karta hai jab hum false karte hai tab yai state ya props ko update hone nahi deta.

componentWillUpdate - This function is generally called before the component is updated or when the state or props passed to the component changes. Don't call setState()method in this function. This method will not be invoked if shouldComponentUpdate() methods return false.

componentDidUpdate = This method works when the state and props are changed.

componentWillUnmount = This method works when we move to one component to other or close the component.

❖ Conditional Rendering

- ➢ There are two ways to create a conditional rendering

```
            <>
                {(this.state.isLogin) ? <> <button onClick={() => { this.setState({ isLogin:
!this.state.isLogin }) }}>Logout</button></> : <> <button onClick={() => { this.setState({
isLogin: !this.state.isLogin }) }}>Login</button></>}
                <> <button onClick={() => { this.setState({ isLogin: !this.state.isLogin })
}}>{(this.state.isLogin) ? "Login" : "Logout"}</button></>

            </>
```

```
        // render() {
        //     if (this.state.isLogin) {

        //         return (<>
        //             {/* <button onClick={()=>{ this.setState({ isLogin:false })
}}>Login</button> */}
        //             <button onClick={() => { this.setState({ isLogin: !this.state.isLogin })
}}>Logout</button>
        //         </>);
        //     }
        //     else {
        //         return (<>
        //             {/* <button onClick={()=>{ this.setState({ isLogin:false })
}}>Login</button> */}
        //             <button onClick={() => { this.setState({ isLogin: !this.state.isLogin })
}}>Login</button>
        //         </>);
        //     }
```

❖ Class List and keys

- Jab list ki bat kare tab hum array ka use karte hai kyoki array is always return index value.
- Keys jab hamare pass key or value ek se jyada hoti hai tab hum usko return karvane ke liye map ka use karte hai yani keys as a object return karega.

❖ Spread and Rest

- Rest and Spread ko (…) dot se use karte hai. But dono alag alag opretor hai.
- Rest always function me as parameter use krate hai.
- Spread always function me as argument use kar te hai. And array ko merge kar ke new array create kar na ho to spread operator use kar sakte hai. Object ko bhi aese merge kar na hoto spread operator use kar sakte hai.

```
const addition = (a, ...b) => {
            console.log("called addition value of a : ", a, "value of b ", b);
        }
        addition(50, 60, 80, 90)
        addition(50, 60)
        const additionSpread = (a, b, c, d) => {
            console.log("called addition value of a : ", a, b, c, d);
        }
        additionSpread(50, 60, 80, 90)
        const numbers = [1, 3, 5, 7]
        additionSpread(...numbers)
        additionSpread(50, 60)
```

❖ Dynamic Sub menu

```
import React, { Component } from "react";
import { Link } from "react-router-dom";
class DynamicSubMenuex extends Component {
    constructor(props) {
        super(props);
        this.state = { open: true };
    }
    inputHandle = () => {
        this.setState({ open: !this.state.open })
    }
    render() {
        const menuItems = [
            {
                title: 'Services',
                url: '/services',
                submenu: [
                    {
                        title: 'web design',
                        url: 'web-design',
                    }, {
                        title: 'web development',
                        url: 'web-dev',
                    }, {
                        title: 'SEO',
                        url: 'seo',
                    },
                ],
            },
        ];
        const Menudata = menuItems.map((data) => {

            let submenudata = data.submenu.map((submenu) => {
                return <>

                    <li><Link to={submenu.url}>{submenu.title}</Link></li>
                </>
            })
            return <>
                {/* <ol>{submenudata}</ol> */}
                {this.state.open ? <>
                    <button onClick={this.inputHandle}>Services</button>
                </> : <>
                    <button onClick={this.inputHandle}>Services</button>
                    <ol>{submenudata}</ol>

                </>}
            </>
        })

        return (<>
            <ol>{Menudata}</ol>
        </>)
    }
}

export default DynamicSubMenuex;
```

❖ Controlled component

- Controlled component always handle by state.
- Jub hame form ke inner data ko controlled karvana ho tub hum controlled component ka use kar sakte hai.

❖ Uncontrolled component

- Uncontrolled component always handle by JavaScript.
- Jub hame form ko handle karvana ho ya fir koi event ke use karke form ke data ko pass on karvana ho tub hum uncontrolled component kar sakte hai.
- When we want to get the reference value from the input and pass on to the component then we use uc.

❖ Composition vs Inheritance

- Jab me koe ek component se dushre component me state ke data ko transfer karvana ho tub hum apne child component mai attribute pass karte hai as a state abhi hame apne child usko receive karvane ke liye props ke dhwara hum receive kar sakte hai.

- Normally we pass the data using props it is working but when we want to pass the child element data it not possible using single props so we want to pass all child data then we use {this.props.children}.

❖ State Lifting

- Subse pehle hum state ko define karte hai.
- Uske bad child component create karenge or attribute ko create karke function define karenge.
- Parent component ke undar hum child component ko load karvaye ge or hum attribute create karke function ko define karenge.
- Pehle hum input create karenge or inside the input tag hum event create karenge or child component mai hum arrow function banayenge or state ke data ko receive karenge.
- Inside the parent component hum data ko receive karenge as a parameter and setState ke dhwara hum us data ko update karenge finally hum p tag mai us state ke change ho rahe data ko store karvange.

- First we define the state.
- In parent component we load child component and we pass attribute and we define function.
- Then in child component create one input text inside textbox pass the onChange event we create arrow function inside child component and receive the state data.
- Now inside the parent component we receive the data as a parameter and update the data using setState whenever we change the input our data will also change inside the p tag.

❖ Higher Order Component

- Jab hame ek component mese dushre component me function bejna ho tabhi hum hoc ka use karte hai.
- Pehle hum button create karenge inside the onclick even hum function ko pass karenge.
- Hum apne Main compo ko wrap karenge Enhanced compo ke sath.
- Then hum Enhanced compo ko as a callback function create karenge or usme hum argument pass karenge.
- Uske undar hum class create karenge or apne argument ko as a component load karenge or usme jo hamne main compo ke undar jo hamne pass kiya tha function usko hum apne component ke undar call karenge.
- Function ko create karenge or hame jo perfrom karna he vo karenge.
- Then hum apne class ko return karenge.