**Ronak Patel**

**DAY -22 TASK**

**Q) Learn about Component Lifecycle methods of React.**

**Mounting**

**Updating**

**Unmounting**

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.The three phases are: **Mounting**, **Updating**, and **Unmounting**.

Mounting:-

Mounting means putting elements into the DOM.

React has four built-in methods that gets called, in this order, when mounting a component:

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

The render() method is required and will always be called, the others are optional and will be called if you define them.

**Constructor:-**

The constructor() method is called before anything else, when the component is initiated, and it is the natural place to set up the initial state and other initial values.The constructor() method is called with the props, as arguments, and you should always start by calling the super(props) before anything else, this will initiate the parent's constructor method and allows the component to inherit methods from its parent (React.Component).

Example :-

*constructor(props){*

*super(props);*

*this.state = {  data: ‘www.facebook.com’ }*

### **getDerivedStateFromProps**

The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM. This is the natural place to set the state object based on the initial props. It takes state as an argument, and returns an object with changes to the state.

Example :-

*class List extends React.Component {*

*static getDerivedStateFromProps(props, state) {*

*if (props.selected !== state.selected) {*

*return {*

*selected: props.selected,*

*};*

*}*

*// Return null if the state hasn't changed*

*return null;*

*}*

*// ...*

*}*

### **render**

The render() method is required, and is the method that actual outputs HTML to the DOM.

*class Header extends React.Component {*

*render() {*

*return (*

*<h1>This is the content of the Header component</h1>*

*);*

*}*

*}ReactDOM.render(<Header />, document.getElementById('root'));*

### **componentDidMount**

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

This is where you run statements that requires that the component is already placed in the DOM.

*import React, { Component } from 'react';*

*class App extends Component {*

*constructor(props){*

*super(props);*

*this.state = {*

*data: 'Jordan Belfort'*

*}*

*}*

*getData(){*

*setTimeout(() => {*

*console.log('Our data is fetched');*

*this.setState({*

*data: 'Hello WallStreet'*

*})*

*}, 1000)*

*}*

*componentDidMount(){*

*this.getData();*

*}*

*render() {*

*return(*

*<div>*

*{this.state.data}*

*</div>*

*)}}*

*export default App;*

## Updating

The next phase in the lifecycle is when a component is updated.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

### **getDerivedStateFromProps**

Also at updates the getDerivedStateFromProps method is called. This is the first method that is called when a component gets updated.This is still the natural place to set the state object based on the initial props.

Example :-

*class List extends React.Component {*

*static getDerivedStateFromProps(props, state) {*

*if (props.selected !== state.selected) {*

*return {*

*selected: props.selected,*

*};*

*}*

### **shouldComponentUpdate**

In the shouldComponentUpdate() method you can return a Boolean value that specifies whether React should continue with the rendering or not.The default value is true.

Example:-

class ListItem extends Component {

shouldComponentUpdate(nextProps, nextState) {

return nextProps.isFavourite != this.props.isFavourite;

}

### **getSnapshotBeforeUpdate**

In the getSnapshotBeforeUpdate() method you have access to the props and state before the update, meaning that even after the update, you can check what the values were before the update. If the getSnapshotBeforeUpdate() method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

Example:-

*getSnapshotBeforeUpdate(prevProps, prevState) {*

*document.getElementById("div1").innerHTML =*

*"Before the update, the favorite was " + prevState.favoritecolor;*

*}*

### **componentDidUpdate**

The componentDidUpdate method is called after the component is updated in the DOM.

componentDidUpdate() is invoked immediately after updating occurs. This method is not called for the initial render.Use this as an opportunity to operate on the DOM when the component has been updated. This is also a good place to do network requests as long as you compare the current props to previous props (e.g. a network request may not be necessary if the props have not changed).

Example:-

*componentDidUpdate(prevProps) {*

*if (this.props.userID !== prevProps.userID) {*

*this.fetchData(this.props.userID);*

*}*

*}*

## Unmounting

The next phase in the lifecycle is when a component is removed from the DOM, or unmounting as React likes to call it.

React has only one built-in method that gets called when a component is unmounted: componentWillUnmount :-

Example :-

*import React, { Component } from 'react';*

*export default class SideMenu extends Component {*

*constructor(props) {*

*super(props);*

*this.state = {*

*};*

*this.openMenu = this.openMenu.bind(this);*

*this.closeMenu = this.closeMenu.bind(this);*

*}*

*componentDidMount() {*

*document.addEventListener("click", this.closeMenu);*

*}*

*componentWillUnmount() {*

*document.removeEventListener("click", this.closeMenu);*

*}*

*openMenu() {*

*}*

*closeMenu() {*

*}*

*render() {*

*return (*

*<div><a href="javascript:void(0)"className = "closebtn" onClick = {this.closeMenu}>*

*×</a>*

*<div>Some other structure</div>*

*</div>*

*);*

*}*

*}*