Component life cycle

In React, components have a lifecycle that defines the various stages a component goes through from its creation to its removal from the DOM.

Class Component Lifecycle:

1. Mounting Phase:

- `constructor()`: This is the constructor of the component, where you initialize state and bind event handlers. It's called before anything else.

- `static getDerivedStateFromProps()`: It's called right before rendering the component. It receives props as an argument and returns an object to update the state or null to indicate no state updates are needed.

- `render()`: This method returns the JSX representation of the component. It's called after `constructor` and `getDerivedStateFromProps`.

- `componentDidMount()`: This method is called immediately after the component is inserted into the DOM. It's often used for data fetching, subscriptions, or manually changing the DOM.

2. Updating Phase:

- `static getDerivedStateFromProps()`: It can also be called during the update phase when props change, similar to the mounting phase.

- `shouldComponentUpdate()`: This method allows you to control whether the component should re-render or not based on new props and state. It's often used for performance optimization.

- `render()`: Re-renders the component with updated props and state.

- `componentDidUpdate()`: Called after the component has been re-rendered. It's commonly used for side-effects, like making AJAX requests based on prop changes.

3. Unmounting Phase:

- `componentWillUnmount()`: This method is called just before the component is removed from the DOM. It's used for cleanup tasks like canceling network requests or removing event listeners.

4. Error Handling Phase (optional):

- `static getDerivedStateFromError()`: Introduced in React 16, this method is used to update component state when an error occurs in any child component.

- `componentDidCatch()`: This method is used to log or handle errors that occur in any child component.

Eg

import React, { Component } from 'react';

class LifecycleExample extends Component {

constructor(props) {

super(props);

this.state = {

count: 0,

};

console.log('Constructor is called');

}

static getDerivedStateFromProps(nextProps, prevState) {

console.log('getDerivedStateFromProps is called');

return null;

}

componentDidMount() {

console.log('componentDidMount is called');

}

shouldComponentUpdate(nextProps, nextState) {

console.log('shouldComponentUpdate is called');

return true; // Return true to allow re-rendering; you can use logic to determine this.

}

componentDidUpdate(prevProps, prevState) {

console.log('componentDidUpdate is called');

}

componentWillUnmount() {

console.log('componentWillUnmount is called');

}

incrementCount = () => {

this.setState({ count: this.state.count + 1 });

};

render() {

console.log('render is called');

return (

<div>

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

<button onClick={this.incrementCount}>Increment Count</button>

</div>

);

}

}

export default LifecycleExample;