ReactJS - Component Life Cycle

In this chapter, we will discuss component lifecycle methods.

Lifecycle Methods

- **componentWillMount** is executed before rendering, on both the server and the client side.
- **componentDidMount** is executed after the first render only on the client side. This is where AJAX requests and DOM or state updates should occur. This method is also used for integration with other JavaScript frameworks and any functions with delayed execution such as **setTimeout** or **setInterval**. We are using it to update the state so we can trigger the other lifecycle methods.
- **componentWillReceiveProps** is invoked as soon as the props are updated before another render is called. We triggered it from **setNewNumber** when we updated the state.
- **shouldComponentUpdate** should return **true** or **false** value. This will determine if the component will be updated or not. This is set to **true** by default. If you are sure that the component doesn't need to render after **state** or **props** are updated, you can return **false** value.
- **componentWillUpdate** is called just before rendering.
- **componentDidUpdate** is called just after rendering.
- **componentWillUnmount** is called after the component is unmounted from the dom. We are unmounting our component in **main.js**.

In the following example, we will set the initial **state** in the constructor function. The **setNewnumber** is used to update the **state**. All the lifecycle methods are inside the Content component.

App.jsx

```
import React from 'react';

class App extends React.Component {
   constructor(props) {
      super(props);

      this.state = {
         data: 0
      }
      this.setNewNumber = this.setNewNumber.bind(this)
   };
   setNewNumber() {
      this.setState({data: this.state.data + 1})
   }
   render() {
      return (
         <div>
            <button onClick = {this.setNewNumber}>INCREMENT</button>
            <Content myNumber = {this.state.data}></Content>
         </div>
      );
   }
}
class Content extends React.Component {
   componentWillMount() {
      console.log('Component WILL MOUNT!')
   }
   componentDidMount() {
      console.log('Component DID MOUNT!')
   }
   componentWillReceiveProps(newProps) {
      console.log('Component WILL RECIEVE PROPS!')
   }
   shouldComponentUpdate(newProps, newState) {
      return true;
   }
   componentWillUpdate(nextProps, nextState) {
      console.log('Component WILL UPDATE!');
   }
   componentDidUpdate(prevProps, prevState) {
      console.log('Component DID UPDATE!')
   }
   componentWillUnmount() {
      console.log('Component WILL UNMOUNT!')
   }
   render() {
      return (
         <div>
            <h3>{this.props.myNumber}</h3>
         </div>
      );
   }
```

```
}
export default App;
```

main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.jsx';

ReactDOM.render(<App/>, document.getElementById('app'));

setTimeout(() => {
   ReactDOM.unmountComponentAtNode(document.getElementById('app'));}, 10000);
```

After the initial render, we will get the following screen.

Previous Page Next Page