Conditional Rendering | Cheat Sheet

Concepts in Focus

1 . Conditional Rendering

Conditional Rendering allows us to render different elements or components based on a condition.

Different ways to implement **Conditional Rendering** are:

- Using an If...Else Statement
- Using Element Variables
- Using Ternary Operators
- Using Logical && Operator

1.1 Using an If...Else Statement

```
import { Component } from "react"
import './App.css'

class App extends Component {
  state = { isLoggedIn: true }

   renderAuthButton = () => {
    const {isLoggedIn} = this.state
    if (isLoggedIn === true) {
      return <button>Logout</button>
    }
    return <button>Login</button>
  }

  render() {
    return (
     <div className="container">
        {this.renderAuthButton()}
      </div>
    )
  }
}

export default App
```

1.2 Using Element Variables

```jsx
import { Component } from "react"
import './App.css'

class App extends Component {
  state = { isLoggedIn: true }

  render() {
    const { isLoggedIn } = this.state
    let authButton
    if (isLoggedIn) {
      authButton = <button>Logout</button>
    } else {
      authButton = <button>Login</button>
    }
    return (
      <div className="container">
        <h1>React JS</h1>
        {authButton}
      </div>
    )
 }
}

export default App
```

1.3 Using Ternary Operators

```jsx
import { Component } from "react"
import './App.css'

class App extends Component {

  render() {
    const { isLoggedIn } = this.state
    return (
      <div className="container">
        {isLoggedIn ? <button>Logout</button> : <button>Login</button>}
      </div>
    )
  }
}

export default App
```

1.4 Using Logical && Operator

```jsx
import { Component } from "react"
import './App.css'

class App extends Component {

  render() {
    const { isLoggedIn } = this.state
    return (
      <div className="container">
        {isLoggedIn && <button>Logout</button>}
        {!isLoggedIn && <button>Login</button>}
      </div>
    )
  }
}
```

```

export default App
```

Note

Conditional Rendering can be achieved using inline styles or adding classes with CSS `display` property with value `none` . However, it is preferable.

2. Default Props

defaultProps is a property in React Component used to set default values for the props. This is similar to adding default parameters to the fun

Syntax:

```
// Component Definition

ComponentName.defaultProps = {
  propName1: "propValue1",
  propName2: "propValue2"
}

// Exporting Component
```

Example:

File: src/Welcome/index.js

```
const Welcome = (props) => {
  const { name, greeting } = props;
  return (
    <h1 className="message">
      {greeting}, {name}
    </h1>
  );
};

Welcome.defaultProps = {
  name: "Rahul",
  greeting: "Hello"
};

export default Welcome;
```

File: src/App.js

```
import { Component } from "react";
import Welcome from "./Welcome";

class App extends Component {
  state = { isLoggedIn: true };
  render() {
    const { isLoggedIn } = this.state;
    return (
      <div className="container">
        <Welcome greeting="Hello" />
      </div>
    );
  }
}

export default App;
```

Note

While accessing the props, the correct prop name should