📌 Complete Example: Conditional Rendering in React

Conditional rendering in React **dynamically shows or hides UI elements** based on application state. This technique is essential for building **interactive UIs, dashboards, authentication flows, and dynamic content updates**.

✔ Features in This Example

- ✅ **Using `if` statements for conditional rendering**
- ✅ **Using the ternary (`? :`) and logical (`&&`) operators**
- ✅ **Hiding and showing components dynamically**
- ✅ **Handling authentication state and error messages**
- ✅ **Loading states for API calls or async operations**

 `ConditionalRendering.js` (Functional Component Example)

This component demonstrates various ways to **conditionally render elements**.

```
jsx
CopyEdit
import React, { useState } from "react";

const ConditionalRendering = () => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [hasError, setHasError] = useState(false);
  const [isLoading, setIsLoading] = useState(false);

  // Simulate login process
  const handleLogin = () => {
    setIsLoading(true);
    setTimeout(() => {
      setIsLoggedIn(true);
      setIsLoading(false);
      setHasError(false);
    }, 2000);
  };

  // Simulate logout
  const handleLogout = () => {
```

```
  setIsLoggedIn(false);
 };

 // Simulate an error
 const triggerError = () => {
  setHasError(true);
 };

 return (
  <div style={styles.container}>
   <h2>Conditional Rendering Example</h2>

   {/* Show loading state */}
   {isLoading && <p style={styles.loading}>🔄 Logging in...</p>}

   {/* Show error message if there's an error */}
   {hasError && <p style={styles.error}>❌ Error: Unable to login!</p>}

   {/* Conditional rendering using if-else */}
   {!isLoggedIn ? (
    <div>
     <p>Please log in to continue.</p>
     <button style={styles.button} onClick={handleLogin}>
      Log In
     </button>
     <button style={styles.button} onClick={triggerError}>
      Simulate Error
     </button>
    </div>
   ) : (
    <div>
     <p>✅ Welcome, user! You are logged in.</p>
     <button style={styles.button} onClick={handleLogout}>
      Log Out
     </button>
    </div>
   )}

   {/* Ternary Operator Example */}
   <p>
    {isLoggedIn ? "🎉 Enjoy your session!" : "🔒 Please log in first."}
   </p>

   {/* Logical && Operator Example */}
   {isLoggedIn && <p>🎁 Special message only for logged-in users!</p>}
  </div>
 );
};

// Inline styles for better visualization
const styles = {
 container: {
```

```
    textAlign: "center",
    padding: "20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    width: "350px",
    margin: "20px auto",
    backgroundColor: "#f9f9f9",
  },
  button: {
    margin: "10px",
    padding: "10px",
    fontSize: "16px",
    cursor: "pointer",
  },
  error: {
    color: "red",
    fontWeight: "bold",
  },
  loading: {
    color: "blue",
    fontWeight: "bold",
  },
};

export default ConditionalRendering;
```

 **ConditionalRenderingClass.js** **(Class Component Example)**

This example **demonstrates conditional rendering in a class component** using this.state.

jsx
CopyEdit

```
import React, { Component } from "react";

class ConditionalRenderingClass extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isLoggedIn: false,
      hasError: false,
      isLoading: false,
    };
  }

  handleLogin = () => {
    this.setState({ isLoading: true });
    setTimeout(() => {
      this.setState({ isLoggedIn: true, isLoading: false, hasError: false });
    }, 2000);
  };
```

```
  handleLogout = () => {
    this.setState({ isLoggedIn: false });
  };

  triggerError = () => {
    this.setState({ hasError: true });
  };

  render() {
    const { isLoggedIn, hasError, isLoading } = this.state;

    return (
      <div style={styles.container}>
        <h2>Conditional Rendering in Class Component</h2>

        {/* Show loading indicator */}
        {isLoading && <p style={styles.loading}>🔄 Logging in...</p>}

        {/* Show error message if an error occurred */}
        {hasError && <p style={styles.error}>❌ Error: Login failed!</p>}

        {/* Conditional rendering using if-else */}
        {!isLoggedIn ? (
          <div>
            <p>Please log in to continue.</p>
            <button style={styles.button} onClick={this.handleLogin}>
              Log In
            </button>
            <button style={styles.button} onClick={this.triggerError}>
              Simulate Error
            </button>
          </div>
        ) : (
          <div>
            <p>✅ Welcome back! You are logged in.</p>
            <button style={styles.button} onClick={this.handleLogout}>
              Log Out
            </button>
          </div>
        )}

        {/* Ternary operator example */}
        <p>{isLoggedIn ? "🎉 Enjoy your session!" : "🔒 Please log in first."}</p>

        {/* Logical AND (&&) Example */}
        {isLoggedIn && <p>🕷️ Special content for logged-in users!</p>}
      </div>
    );
  }
}
```

```
const styles = {
  container: {
    textAlign: "center",
    padding: "20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    width: "350px",
    margin: "20px auto",
    backgroundColor: "#f9f9f9",
  },
  button: {
    margin: "10px",
    padding: "10px",
    fontSize: "16px",
    cursor: "pointer",
  },
  error: {
    color: "red",
    fontWeight: "bold",
  },
  loading: {
    color: "blue",
    fontWeight: "bold",
  },
};

export default ConditionalRenderingClass;
```

App.js (Parent Component to Display Both Examples)

This file renders both **functional and class components** demonstrating conditional rendering.

```
jsx
CopyEdit
import React from "react";
import ConditionalRendering from "./ConditionalRendering"; // Functional Component
import ConditionalRenderingClass from "./ConditionalRenderingClass"; // Class Component

const App = () => {
  return (
    <div className="App">
      <h1>React Conditional Rendering</h1>
      <ConditionalRendering />
      <ConditionalRenderingClass />
    </div>
  );
};

export default App;
```

✔ How Conditional Rendering Works

Method	Example
Using if Statements	if (condition) { return <Component />; }
Ternary Operator (? :)	{condition ? <Component1 /> : <Component2 />}
Logical AND (&&)	{condition && <Component />}

🚀 Real-World Use Cases

- **Authentication**: Show login button if user is logged out; show logout button if logged in.
- **Loading States**: Show "Loading..." message while fetching data from an API.
- **Error Handling**: Display error messages when form submission fails.
- **Feature Flags**: Enable/disable features based on user roles or permissions.

🔍 Key Features Explained:

✔ **Functional Component** → Uses an arrow function instead of a class.
✔ **React Hook (useState)** manages three different state variables.
✔ **Conditional Rendering:**

- if-else inside JSX → Displays different UI based on isLoggedIn.
- **Ternary Operator** → isLoggedIn ? "Welcome" : "Please log in".
- **Logical AND (&&) Operator** → isLoggedIn && Special Message .

✔ **Event Handling in React:**

- handleLogin() → Simulates a login with a delay.
- handleLogout() → Resets login state.
- triggerError() → Simulates an error.

✔ **JSX Rendering:** Updates UI dynamically when state changes.
✔ **Inline Styling:** Defined in a styles object and applied using style attribute.
✔ **Component Export:** export default ConditionalRendering; makes it reusable.