Day 4: Conditional Rendering & Lists in React

Today, you'll learn how to **conditionally render** elements in React and work with **lists** by dynamically rendering content.

---

1 Conditional Rendering in React

React provides multiple ways to render elements conditionally based on state or props.

Using If-Else Statement

You can use a regular `if` statement to decide what to render. However, for conditional rendering inside JSX, you'll need to use ternary operators or logical operators.

```
function WelcomeMessage({ isLoggedIn }) {
  if (isLoggedIn) {
    return <h1>Welcome back, user!</h1>;
  } else {
    return <h1>Please log in.</h1>;
  }
}

export default WelcomeMessage;
```

---

2 Using Ternary Operator for Conditional Rendering

Ternary operator provides a shorthand way to write if-else. It's useful for inline conditions.

Example:

```
function WelcomeMessage({ isLoggedIn }) {
  return (
    <h1>
      {isLoggedIn ? "Welcome back, user!" : "Please log in."}
    </h1>
  );
}

export default WelcomeMessage;
```

✅ **Ternary operator** allows you to choose between two options.

3️⃣ Logical AND (&&) Operator for Conditional Rendering

The logical && operator is often used when you want to render something **only if the condition is true**. If false, nothing will render.

Example:

```
function Notifications({ unreadMessages }) {
  return (
    <div>
      {unreadMessages > 0 && <h2>You have {unreadMessages} new
messages!</h2>}
    </div>
  );
}

export default Notifications;
```

✅ **This only displays the message if there are unread messages.**

4️⃣ Rendering Lists in React

Rendering a **list of items** is common in React apps. Use JavaScript's `map()` function to loop over an array of items and return a list of components or elements.

Basic List Rendering

```
function ItemList() {
  const items = ["Apple", "Banana", "Cherry"];

  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}

export default ItemList;
```

✅ **key helps React optimize rendering by identifying elements uniquely.**

5️⃣ Handling Lists with Objects

If your list contains objects, you can access properties inside the `map()` function.

Example:

```
function ProductList() {
  const products = [
    { id: 1, name: "Laptop", price: "$999" },
    { id: 2, name: "Smartphone", price: "$599" },
    { id: 3, name: "Tablet", price: "$399" },
  ];

  return (
    <ul>
      {products.map((product) => (
        <li key={product.id}>
          {product.name} - {product.price}
        </li>
      ))}
    </ul>
  );
}

export default ProductList;
```

✅ **You can dynamically render complex data using lists of objects.**

---

6 Conditional Rendering with Lists

You can combine **conditional rendering** and **lists** to display items based on certain conditions.

Example:

```
function TaskList() {
  const tasks = [
    { id: 1, name: "Learn React", completed: true },
    { id: 2, name: "Build a project", completed: false },
    { id: 3, name: "Read a book", completed: true },
  ];

  return (
    <ul>
      {tasks.map((task) => (
        task.completed && <li key={task.id}>{task.name}</li>
      ))}
    </ul>
  );
}

export default TaskList;
```

✅ **Only completed tasks are rendered.**

📌 Summary of Day 4

- ✔ **Conditional Rendering** with `if`, ternary, and logical operators
- ✔ Render lists using `map()` function
- ✔ Handle **lists with objects** in JSX
- ✔ Combine conditional rendering with lists

🚀 Next Step: Day 5 - Handling Forms & Controlled Components