📘 Introduction to Custom Higher-Order Components (HOCs) in React

A **Higher-Order Component (HOC)** is an advanced **pattern in React** that allows you to **reuse component logic**. It's a function that **takes a component and returns a new component** with enhanced behavior.

🔧 Basic Syntax of a Custom HOC

jsx Copy Edit

```jsx
function withExtraInfo(WrappedComponent) { return function EnhancedComponent(props) { return <WrappedComponent {...props} extraInfo="Hello from HOC!" />; }; }
```

✅ Example: Custom HOC to Add Loading State

Step 1: Create the HOC

jsx Copy Edit

```jsx
function withLoading(Component) { return function WrappedComponent({ isLoading, ...rest }) { if (isLoading) { return <p>Loading...</p>; } return <Component {...rest} />; }; }
```

Step 2: Create a Component to Wrap

jsx Copy Edit

```jsx
function UserList({ users }) { return ( <ul> {users.map((user) => <li key={user.id}>{user.name} </li>)} </ul> ); }
```

Step 3: Use the HOC

jsx Copy Edit

```jsx
const UserListWithLoading = withLoading(UserList);
```

Step 4: Render the Enhanced Component

jsx Copy Edit

```jsx
<UserListWithLoading isLoading={true} users={[]} />
```

🧠 Why Use HOCs?

- Reuse logic like **loading states**, **authentication**, or **data fetching**.
- Keep components clean and focused.
- Compose behavior without changing the original component.

🚫 HOC Limitations

- **Can become hard** to trace with deeply nested HOCs.
- **Less preferred now** compared to **hooks** for many use cases.