✅ Step-by-Step: Create a `withLoading` HOC

1. Create the HOC

jsx Copy Edit

```jsx
// withLoading.js import React from 'react'; function withLoading(Component) { return function WithLoadingComponent({ isLoading, ...props }) { if (isLoading) { return <div>Loading...</div>; // Or use a spinner component here } return <Component {...props} />; }; } export default withLoading;
```

2. Create a Wrapped Component

jsx Copy Edit

```jsx
// UserList.js import React from 'react'; function UserList({ users }) { return ( <ul> {users.map((user) => ( <li key={user.id}>{user.name}</li> ))} </ul> ); } export default UserList;
```

3. Wrap the Component with HOC

jsx Copy Edit

```jsx
// App.js import React, { useState, useEffect } from 'react'; import UserList from './UserList'; import withLoading from './withLoading'; const UserListWithLoading = withLoading(UserList); function App() { const [users, setUsers] = useState([]); const [isLoading, setIsLoading] = useState(true); // Simulate data fetching useEffect(() => { setTimeout(() => { setUsers([ { id: 1, name: 'Alice' }, { id: 2, name: 'Bob' } ]); setIsLoading(false); }, 2000); }, []); return ( <div> <h1>User List</h1> <UserListWithLoading isLoading={isLoading} users={users} /> </div> ); } export default App;
```

📝 Summary

- `withLoading` is a **Higher-Order Component** that wraps any component to show a loading state.
- The real component is only rendered once `isLoading` is false.
- This promotes **code reuse** and **separation of concerns**.