import React, { Component } from "react";

class ContactList extends React.Component {

  render() {

    const people = this.props.contacts

    return <ol>

      {people.map((person) =>(

        <li key={person.name}> {person.name}</li>

      ))}

    </ol>

  }

}

class App extends Component {

  render() {

   return (

    <div className="App">

      <ContactList contacts={[

      { name: 'Tyler'},

      {name: 'Karen'},

      {name: 'Richard'}

    ]}/>

      <ContactList contacts={[

        {name: 'Amanda'},

        {name: 'Mikenzi'},

        {name: 'Ryan'}

      ]}/>

    </div>

  );

 }

}

export default App;

[Here's the commit with the changes made in this video.](https://github.com/udacity/reactnd-contacts-app/commit/f1fbeff5533d82e33b075573fff684a99a82b845)

**Favor Composition Over Inheritance**

You might have heard before that it’s better to “favor composition over inheritance”. This is a principle that I believe is difficult to learn today. Many of the most popular programming languages make extensive use of inheritance, and it has carried over into popular UI frameworks like the Android and iOS SDKs.

In contrast, React uses composition to build user interfaces. Yes, we extend React.Component, but we never extend it more than once. Instead of extending base components to add more UI or behavior, we compose elements in different ways using nesting and props. You ultimately want your UI components to be independent, focused, and *reusable*.

So if you’ve never understood what it means to “favor composition over inheritance” you’ll definitely learn using React!