React 拥有一个强大的组合模型，我们建议使用组合而不是继承以实现代码的重用。

**包含**

一些组件在设计前无法获知自己要使用什么子组件

我们建议这种组件使用特别的 children prop 来直接传递子元素到他们的输出中：

function FancyBorder(props) {

return (

<div className={'FancyBorder FancyBorder-' + props.color}>

{props.children}

</div>

);

}

这允许其他组件通过嵌套 JSX 传递任意子组件给他们：

function WelcomeDialog() {

return (

<FancyBorder color="blue">

<h1 className="Dialog-title">Welcome</h1>

<p className="Dialog-message">Thank you for visiting our spacecraft!

</p>

</FancyBorder>

);

}

有时候，在一个组件中你可能需要多个 “占位符”。在这种情况下，你可以使用自定义的 prop(属性)，而不是使用 children ：

function SplitPane(props) {

return (

<div className="SplitPane">

<div className="SplitPane-left">

{props.left}

</div>

<div className="SplitPane-right">

{props.right}

</div>

</div>

);

}

function App() {

return (

<SplitPane

left={ <Contacts /> }

right={ <Chat /> }

/>

);

}

**如何看待继？**

在 Facebook ，我们在千万的组件中使用 React，我们还没有发现任何用例，值得我们建议你用继承层次结构来创建组件。

使用 props(属性) 和 组合已经足够灵活来明确、安全的定制一个组件的外观和行为。切记，组件可以接受任意的 props(属性) ，包括原始值、React 元素，或者函数。

如果要在组件之间重用非 U I功能，我们建议将其提取到单独的 JavaScript 模块中。组件可以导入它并使用该函数，对象或类，而不扩展它。