在 React 中，你可以创建不同的组件封装你所需要的行为。然后，只渲染它们之中的一些，取决于你的应用的状态。

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

if (isLoggedIn) {

return <UserGreeting />;

}

return <GuestGreeting />;

}

ReactDOM.render(

<Greeting isLoggedIn={false} />,

document.getElementById('root'));

### 使用逻辑 && 操作符的内联 if 用法

您可以 [在JSX中嵌入任何表达式](http://www.css88.com/react/docs/introducing-jsx.html" \l "embedding-expressions-in-jsx) ，方法是将其包裹在花括号中。这也包括 JavaScript 逻辑 && 运算符。 它有助于有条件地包含一个元素：

function Mailbox(props) {

const unreadMessages = props.unreadMessages;

return (

<div>

<h1>Hello!</h1>

{unreadMessages.length > 0 &&

<h2>

You have {unreadMessages.length} unread messages.

</h2>

}

</div>

);

}

如果条件为 true ，则 && 后面的元素将显示在输出中。 如果是 false，React 将会忽略并跳过它。

### 使用条件操作符的内联 If-Else

在下面这个例子中，我们使用它来进行条件渲染一个小的文本块：

render() {

const isLoggedIn = this.state.isLoggedIn;

return (

<div>

The user is <b>{isLoggedIn ? 'currently' : 'not'}</b> logged in.

</div>

);

}

它也可以用于更大的表达式，虽然不太明显发生了什么：

render() {

const isLoggedIn = this.state.isLoggedIn;

return (

<div>

{isLoggedIn ? (

<LogoutButton onClick={this.handleLogoutClick} />

) : (

<LoginButton onClick={this.handleLoginClick} />

)}

</div>

);

}

### 防止组件渲染

在极少数情况下，您可能希望组件隐藏自身，即使它是由另一个组件渲染的。为此，返回 null 而不是其渲染输出。

在下面的例子中，根据名为warn的 prop 值，呈现 <WarningBanner /> 。如果 prop 值为 false ，则该组件不渲染：

function WarningBanner(props) {

if (!props.warn) {

return null;

}

return (

<div className="warning">

Warning!

</div>

);

}