React 事件处理 React 列表 & Keys

React 条件渲染

在 React 中，你可以创建不同的组件来封装各种你需要的行为。然后还可以根据应用的状态变化只渲染其中的一部分。

React 中的条件渲染和 JavaScript 中的一致，使用 JavaScript 操作符 if 或条件运算符来创建表示当前状态的元素，然后让 React 根据它们来更新 UI。

先来看两个组件:

```
function UserGreeting(props) {
  return <h1>欢迎回来!</h1>;
}

function GuestGreeting(props) {
  return <h1>请先注册。</h1>;
}
```

我们将创建一个 Greeting 组件，它会根据用户是否登录来显示其中之一：

React 实例

```
function Greeting(props) {
const isLoggedIn = props.isLoggedIn;
if (isLoggedIn) {
return <UserGreeting />;
}
return <GuestGreeting />;
}
ReactDOM.render(
// 尝试修改 isLoggedIn={true}:
<Greeting isLoggedIn={false} />,
document.getElementById('example')
);
```

尝试一下 »

元素变量

你可以使用变量来储存元素。它可以帮助你有条件的渲染组件的一部分，而输出的其他部分不会更改。

在下面的例子中，我们将要创建一个名为 LoginControl 的有状态的组件。

它会根据当前的状态来渲染 `<LoginButton />` 或 `<LogoutButton />`，它也将渲染前面例子中的 `<Greeting />`。

React 实例

```
class LoginControl extends React.Component {
constructor(props) {
super(props);
this.handleLoginClick = this.handleLoginClick.bind(this);
```

```
this.handleLogoutClick = this.handleLogoutClick.bind(this);
this.state = {isLoggedIn: false};
}
handleLoginClick() {
this.setState({isLoggedIn: true});
}
handleLogoutClick() {
this.setState({isLoggedIn: false});
}
render() {
const isLoggedIn = this.state.isLoggedIn;
let button = null;
if (isLoggedIn) {
button = <LogoutButton onClick={this.handleLogoutClick} />;
} else {
button = <LoginButton onClick={this.handleLoginClick} />;
}
return (
<div>
<Greeting isLoggedIn={isLoggedIn} />
{button}
</div>
);
}
}
ReactDOM.render(
<LoginControl />,
document.getElementById('example')
);
```

尝试一下 »

与运算符 &&

你可以通过用花括号包裹代码在 JSX 中嵌入任何表达式 ，也包括 JavaScript 的逻辑与 &&，它可以方便地条件渲染一个元素。

React 实例

```
function Mailbox(props) {
const unreadMessages = props.unreadMessages;
return (
<div>
<h1>Hello!</h1>
{unreadMessages.length > 0 &&
<h2>
您有 {unreadMessages.length} 条未读信息。
</h2>
}
</div>
);
}
const messages = ['React', 're: React', 'Re:Re: React'];
ReactDOM.render(
```

```
<Mailbox unreadMessages={messages} />,
document.getElementById('example')
);
```

尝试一下 »

在 JavaScript 中，`true && expression` 总是返回 **expression**，而 `false && expression` 总是返回 **false**。

因此，如果条件是 **true**，`&&` 右侧的元素就会被渲染，如果是 **false**，React 会忽略并跳过它。

三目运算符

条件渲染的另一种方法是使用 JavaScript 的条件运算符:

```
condition ? true : false。
```

在下面的例子中，我们用它来有条件的渲染一小段文本。

render() { const isLoggedIn = this.state.isLoggedIn; return (The user is **{isLoggedIn ? 'currently' : 'not'}** logged in.); } 同样它也可以用在较大的表达式中，虽然不太直观：

```
render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      {isLoggedIn ? (
        <LogoutButton onClick={this.handleLogoutClick} />
      ) : (
        <LoginButton onClick={this.handleLoginClick} />
      )}
    </div>
  );
}
```

阻止组件渲染

在极少数情况下，你可能希望隐藏组件，即使它被其他组件渲染。让 render 方法返回 null 而不是它的渲染结果即可实现。

在下面的例子中，`<WarningBanner />` 根据属性 warn 的值条件渲染。如果 warn 的值是 false，则组件不会渲染：

React 实例

```
function WarningBanner(props) {
if (!props.warn) {
return null;
}
return (
<div className="warning">
警告!
</div>
```

```
);
}
class Page extends React.Component {
constructor(props) {
super(props);
this.state = {showWarning: true}
this.handleToggleClick = this.handleToggleClick.bind(this);
}
handleToggleClick() {
this.setState(prevState => ({
showWarning: !prevState.showWarning
}));
}
render() {
return (
<div>
<WarningBanner warn={this.state.showWarning} />
<button onClick={this.handleToggleClick}>
{this.state.showWarning ? '隐藏' : '显示'}
</button>
</div>
);
}
}
ReactDOM.render(
<Page />,
document.getElementById('example')
);
```

尝试一下 »

组件的 render 方法返回 null 并不会影响该组件生命周期方法的回调。例如，componentWillUpdate 和 componentDidUpdate 依然可以被调用。

React 事件处理 React 列表 & Keys

2 篇笔记

写笔记

通过条件渲染页面：

首先建一个函数，来根据不同的情况返回不同的值，然后建一个类组建，先进行变量的初始化，对变量进行操作，在组件内进行小的渲染，最后通过 **ReactDOM.render()** 渲染到页面上。

为什么要进行变量的初始化？

一个软件所分配到的空间中极可能存在着以前其他软件使用过后的残留数据，这些数据被称之为垃圾数据。所以通常情况下我们为一个变量，为一个数组，分配好存储空间之后都要对该内存空间初始化。

简单来说就是清理残留数据。

亮仔 8个月前 (07-22)

```
handleToggleClick(){
  this.setState(prevState=>({ // 这里 prevState
      showWarning:!prevState.showWarning
    })
  );
}
```

以上代码 prevState 如何传递？

setState() 可以接收一个函数，这个函数接受两个参数，第一个参数表示上一个状态值，第二参数表示当前的 props：

```
this.setState((prevState, props) => ({
    //do something here
}));
```

羊吃饱了 1个月前 (02-13)