HTML：

<button onclick="activateLasers()"> Activate Lasers</button>

在 React 中略有不同：

<button onClick={activateLasers}> Activate Lasers</button>

另一个区别是，在 React 中你不能通过返回 false. 必须明确调用 preventDefault 。

例如，对于纯 HTML ，要阻止链接打开一个新页面的默认行为，可以这样写：

<a href="#" onclick="console.log('The link was clicked.'); return false">Click me</a>

在 React 中, 应该这么写:

function ActionLink() {

function handleClick(e) {

e.preventDefault();

console.log('The link was clicked.');

}

return (

<a href="#" onClick={handleClick}> Click me </a>

);

}

这里， e 是一个合成的事件。

当使用一个 [ES6 类](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes) 定义一个组件时，通常的一个事件处理程序是类上的一个方法。例如， Toggle 组件渲染一个按钮，让用户在 “ON” 和 “OFF” 状态之间切换：

class Toggle extends React.Component {

constructor(props) {

super(props);

this.state = {isToggleOn: true};

// 这个绑定是必要的，使`this`在回调中起作用

this.handleClick = this.handleClick.bind(this);

}

handleClick() {

this.setState(prevState => ({ isToggleOn: !prevState.isToggleOn}));

}

render() {

return (

<button onClick={this.handleClick}> {

this.state.isToggleOn ? 'ON' : 'OFF'}

</button>

);

}}

ReactDOM.render(

<Toggle />,

document.getElementById('root'));

在JSX回调中你必须注意 this 的指向。 在 JavaScript 中，类方法默认没有 [绑定](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_objects/Function/bind) 的。如果你忘记绑定 this.handleClick 并将其传递给onClick，那么在直接调用该函数时，this会是 undefined 。

## 将参数传递给事件处理程序

在循环内部，通常需要将一个额外的参数传递给事件处理程序。 例如，如果 id 是一个内联 ID，则以下任一方式都可以正常工作：

<button onClick={(e) => this.deleteRow(id, e)}>Delete Row</button>

<button onClick={this.deleteRow.bind(this, id)}>Delete Row</button>

上述两行代码是等价的，分别使用