React 是用于构建用户界面的一个声明性、高效且灵活的 JavaScript 库。它允许您从称为"组件"的小型和孤立的代码片段中编写复杂的 UIs。

React 有几个不同类型的组件，但我们从子类开始：React.Component

class ShoppingList extends React.Component {

render() {

return (

<div className="shopping-list">

<h1>Shopping List for {this.props.name}</h1>

<ul>

<li>Instagram</li>

<li>WhatsApp</li>

<li>Oculus</li>

</ul>

</div>

);

}

}

// Example usage: <ShoppingList name="Mark" />

我们很快就会找到有趣的XML标签。我们使用组件告诉 React 我们希望在屏幕上看到的内容。当我们的数据发生更改时，React 将有效地更新和重新呈现我们的组件。

在这里，ShoppingList 是一个**响应组件类**，或**React 组件类型**。组件采用参数（"属性"的短），并返回要通过 方法显示的视图层次结构。propsrender

该方法返回要在屏幕上查看的内容的说明。反应采取描述并显示结果。特别是，返回**React 元素**，它是要呈现的内容的轻量级描述。大多数 React 开发人员使用称为"JSX"的特殊语法，这使得这些结构更易于编写。语法在生成时转换为 。上述示例等效于：renderrender<div />React.createElement('div')

return React.createElement('div', {className: 'shopping-list'},

React.createElement('h1', /\* ... h1 children ... \*/),

React.createElement('ul', /\* ... ul children ... \*/)

);

如果您好奇，将在 API 参考中更详细地[描述](https://reactjs.org/docs/react-api.html#createelement)，但我们在本教程中不会使用它。相反，我们将继续使用 JSX。createElement()

JSX 具有 JavaScript 的全部功能。您可以将任何JavaScript表达式放在 JSX 中的大括号中。每个 React 元素都是一个 JavaScript 对象，您可以将其存储在变量中或在程序中传递。

上面的组件仅呈现内置的 DOM 组件，如 和 。但是，您也可以撰写和呈现自定义 React 组件。例如，我们现在可以通过编写 来参考整个购物清单。每个反应组件都封装在一起，可以独立运行;这允许您从简单组件构建复杂的 UIs。ShoppingList<div /><li /><ShoppingList />