# 第二章 JSX

在React中，组件是用于分离关注点的，而不是被当作模板或处理显示逻辑的。在使用React时。你必须接受这样的事实，那就是HTML标签以及生成这些标签的代码之间存在着内在的紧密联系。该设计允许你在构建标签结构时充分利用JavaScript的强大能力，而不必在笨拙的模板语言上浪费时间。

React包含了一种可选的类HTML标记语言。不过在继续之前，我们要先说清楚一件事-------对于那些厌恶在javaScript中写HTML标签以及那些还不明确JSX用处的人，请考虑在React中使用JSX的下列好处：

A. 允许使用使用熟悉的语法来定义HTML元素树。

B. 允许更加语义化且易懂的标签。

C. 程序结构更容易被直观化。

D. 抽象了React Element的创建过程。

E. 可以随时掌控HTML标签以及生成这些标签的代码。

F. 是原生的JavaScript。

在本章里我们会探索JSX的诸多优点，如何使用JSX，以及将它与HTML区分开来的一些注意事项。记住，JSX并不是必需的。如果你决定不使用它，则可以直接跳到本章结尾了解在React中不使用JSX的一些小提示。

## 什么是JSX

JSX即JavaScript XML------一种在React组件内部构建标签的类XML语法。React在不使用JSX的情况一样可以工作，然而使用JSX可以提高组件的可读性，因此推荐你使用JSX。

举个例子，在不使用JSX的React程序中创建一个标题的函数调用大概是这个样子：

//v0.11

React.DOM.h1({className:’quest’},’Questions’);

//v0.12

React.createElement(‘h1’,{className:’question’},’Questions’);

如果使用了JSX，上述调用就变成了下面这种更熟悉且且简练的标签：

<h1 className=”question”>Questions</h1>

与以往在JavaScript中嵌入HTML标签的几种方案相比，JSX有以下几点明显特征：

1、JSX是一种语法变换------每一个JSX节点对应着一个JavaScript函数。

2、JSX即不提供也不需要运行时库。

3、JSX并没有改变或添加javascript的语义------它只是简单的函数调用而已。

与html的相似之处赋予了JSX在React中强大的表现力。下面我们将要讨论使用JSX的好处以及它在程序中发挥的作用，同时还会讨论JSX与HTML的关键区别。

## 使用JSX的好处

当讨论JSX时，很多人会问到的问题就是-------为什么要用它，为什么在已经有那么多模板语言的情况下还要使用JSX，为什么不直接使用原生的javaScript。毕竟，JSX最后只是被简单地转换成对应的javaScript函数而已。

使用JSX有很多好处，而且这些好处会随着代码库的日益增大、组件的愈加复杂而变得越来越明显。我们来看看这些好处究竟是什么。

### 更加熟悉

许多团队包括了非开发人员，例如熟悉HTML的UI及UX设计师和负责完整测试产品的质量保证人员。使用JSX之后，这些团队成员都可以更轻松地阅读和贡献代码。任何熟悉基于XML语言的人都能轻松地掌握JSX。

此外，由于React组件囊括了所有可能的DOM表现形式，因此JSX能巧妙地用简单明了的方式来展现这种结构。

### 更加语义化

除了更加熟悉外，JSX还能够将JavaScript代码转化为更加语义化、更加有意义的标签。这种设计为我们提供了使用类HTML语法来声明组件和数据流向的能力，我们知道它们后续转换为原生的JavaScript。

JSX允许你在应用程序中使用所有预定义的HTML5标签及自定义组件。稍后会讲述更多关于自定义组件的内容，而这里只是简单地说明JSX是如何做到让JavaScript更具有可读性的。

举个例子，让我们设想一个Divider元素，它会渲染出一个位于左边的标题和一个撑满右边的水平分割线。这个Divider的HTML结构大概是下面的样子：

<div className=’divider’>

<h2>Questions</h2><hr />

</div>

把上面的HTML包裹进一个Divider React Component后，你就可以像使用其他任何HTML元素一样使用它了。相比原生的HTML，Diveider提供了更丰富的语义。

<Divider>Questions</Divder>

### 更加直观