# 第六章 组件的复合

在传统HTML当中，元素是构成页面的基础单元。但在React中，构建页面的基础单元是React组件。你可以把React组件理解成混入了Javascript表达能力的HTML元素。实际上写React代码就是构建组件，就像编写HTML文档时使用元素一样。

因为整个React应用都是用组件来构建的，因此这本书完全可以写成一本关于React组件的书。但是本章不会涵盖组件的每一个方面，只介绍一个特性——组件的复合能力。

本质上，一个组件就是一个JavaScript函数，它接受属性（props）和状态（state）作为参数，并输出渲染好的HTML。组件一般被用来呈现和表达应用的某部分数据，因此你可以把React组件理解为HTML元素的扩展。

## 扩展HTML

React+JSX是强大而富有表现力的工具，允许我们使用类似HTML的语法创建自定义元素。比起单纯的HTML，它们还能够控制生命周期中的行为。这些都是从React.createClass方法开始的。

相较于继承，React偏爱复合，即通过结合小巧的，简单的组件和数据对象，构造大而复杂的组件。如果你熟悉其他的MVC或者面向对象工具，你很可能会期望有一个React.extendClass方法可用。然而，正如在构建网页时不会扩展HTML DOM节点那样，React组件是不可以扩展的，而是通过组件之间的组合来构建应用。

React信奉可组合性，你可以混合搭配各种子组件来构成复杂且强大的新组件。举个例子，我们会考虑用户会怎样回答一个调查问卷的问题。特别看一下负责渲染一个选择题，并获取用户答案的AnswerMultipleChoiceQuestion组件。

显然问卷是基于基础的HTML表单元素制作的。要通过封装默认的HTML input元素和定制它们的行为来制作这套问卷回答组件。

## 组件复合的例子

一个渲染选择题的组件要满足以下几个条件：

A.接收一组选项作为输入。

B.把选项渲染给用户。

C.只允许用户选择一个选项。

HTML提供了一些基本的元素——单选类型的输入框和表单组（input group），可以在这里使用。组件的层级从上往下看是这样的：

MultipleChoice->RadioInput->Input(type=”radio”)

这些箭头表示“有一个”。选择题组件MulitpleChoice “有一个”单选框RadioInput，单选框RadioInput“有一个”输入框元素Input。这是组合模式的特征。

## 组装HTML

让我们从下往上开始组装这个组件。React在React.DOM.input的命名空间预定义了input组件，因此我们要做的第一件事情就是把它封装进一个RadioInput组件。这个组件负责定制原本通用的input，将其精缩成与单选按钮行为一致的组件。在对应的示例应用中将其命名为AnswerRadioInput。

先建立一个脚手架，其中包含所需要的渲染方法和基本的标记，用以描述想输出的界面。组件模式开始显示，组件变成了特定类型的输入框。

Var AnswerRadioInput=React.createClass({

Render:function(){

Return (

<div className=”radio”>

<label>

<input type=”radio” />

Label Text

<label/>

)

}

});

### 添加动态属性

现在input还没有内容是动态的，所以下一步需要定义父元素必须传给单选框的那些属性。

A. 这个输入框代表什么值或者选项？

B. 用什么文本来描述它？

C. 这个输入框的name是什么？

D. 也许需要自定义id

E. 也许要重载它的默认值。

有了上述列表以后我们就可以定义这个自定义input的属性类型了。我们把这些添加到类的PropTypes对象当中。

var AnswerRadioInput=React.createClass({

propTypes:{

id:React.PropTypes.string,

name:React.PropTypes.String.isRequired,

label:React.PropTypes.string.isRequired,

value:React.PropTypes.string.isRequired,

checked:React.PropTypes.bool

},

…

});

对于每个非必需的属性我们需要为其定义一个默认值。被它们添加到getDefalultProps方法当中。在每个新的实例当中，如果父组件没有提供给它们的数值，这些值就会被使用。

由于这个方法只会在类上调用一次，而不是在每个实例上调用，因此不能在这里提供id——

每个实例应该保持id的唯一性。这个问题可以用接下来要讲的状态（state）来解决。

Var AnswerRadioInput=React.createClass({

propTypes: {…},

getDefaultProps:function(){

return{

id:null,

checked:false

};

}

});

追踪状态

我们的组件需要记录随时间而变化的数据。尤其是对于每个实例来说都要求是唯一的id，以及用户随时更新的checked值。那么我们来定义初始化状态。

Var AnswerRadioInput=React.createClass({

propTypes:{…},

getDefaultProps:function(){…},

getInitialState:function(){

var id=this.props.id?this.props.id:uniqueId(‘radio-’);

return {

checked:!!this.props.checked,

id:id,

name:id

};

},

…

});

现在你可以更新渲染标记，获取新动态的状态和属性了。

Var AnswerRadio=React.createClass({

propTypes:{…},

getDefaultProps:function(){…},

getInitialState:function(){…},

render:function(){

<div className=”radio”>

<label htmlFor={this.props.id}>

<input type=”radio”

Name={this.props.name}

Id={this.props.id}

Value={this.props.value}

Checked={this.state.checked}

/>

{this.props.label}

</label>

</div>

}

});

### 整合到父组件当中

现在这个组件已经足够完善，可以用到一个父组件中了。接下来我们构建下一层——AnswerMultipleChoiceQuestion。这一层的主要作用是渲染一列选项让用户从中选择。按照上面介绍的模式，我们来创建这个组件基本的HTML和默认属性。

Var AnswerMultipleChoiceQuestion=React.createClass({

propTypes:{

value:React.PropTypes.String,

choices:React.PropTypes.array.isRequired,

onCompleted:React.PropTypes.func.isRequired

},

getInitialState:function(){

return {

id:uniqueId(‘multiple-choice-’),

value:this.props.value

};

},

})