React.js 小书

<-- 返回首页

组件的 render 方法

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson7

React.js 中一切皆组件，用 React.js 写的其实就是 React.js 组件。我们在编写 React.js 组件的时候，一般都需要继承 React.js 的 `Component`（还有别的编写组件的方式我们后续会提到）。一个组件类必须要实现一个 `render` 方法，这个 `render` 方法必须要返回一个 JSX 元素。但这里要注意的是，必须要用一个外层的 JSX 元素把所有内容包裹起来。返回并列多个 JSX 元素是不合法的，下面是错误的做法：

```
...
render () {
  return (
    <div>第一个</div>
    <div>第二个</div>
  )
}
...
```

必须要用一个外层元素把内容进行包裹：

```
...
render () {
  return (
    <div>
      <div>第一个</div>
      <div>第二个</div>
    </div>
  )
}
...
```

表达式插入

在 JSX 当中你可以插入 JavaScript 的表达式，表达式返回的结果会相应地渲染到页面上。表达式用 `{}` 包裹。例如：

```
...
render () {
  const word = 'is good'
  return (
    <div>
      <h1>React 小书 {word}</h1>
    </div>
  )
}
...
```

页面上就显示“React 小书 is good”。你也可以把它改成 `{1 + 2}`，它就会显示“React 小书 3”。你也可以把它写成一个函数表达式返回：

```
...
render () {
  return (
    <div>
      <h1>React 小书 {(function () { return 'is good'})()}</h1>
    </div>
  )
}
...
```

简而言之，`{}` 内可以放任何 JavaScript 的代码，包括变量、表达式计算、函数执行等等。`render` 会把这些代码返回的内容如实地渲染到页面上，非常的灵活。

表达式插入不仅仅可以用在标签内部，也可以用在标签的属性上，例如：

```
...
render () {
  const className = 'header'
  return (
    <div className={className}>
      <h1>React 小书</h1>
    </div>
  )
}
...
```

这样就可以为 `div` 标签添加一个叫 `header` 的类名。

注意，直接使用 `class` 在 React.js 的元素上添加类名如 `<div class=“xxx”>` 这种方式是不合法的。因为 `class` 是 JavaScript 的关键字，所以 React.js 中定义了一种新的方式：`className` 来帮助我们给元素添加类名。

还有一个特例就是 `for` 属性，例如 `<label for='male'>Male</label>`，因为 `for` 也是 JavaScript 的关键字，所以在 JSX 用 `htmlFor` 替代，即 `<label`

`htmlFor='male'>Male</label>`。而其他的 HTML 属性例如 `style` 、`data-*` 等就可以像普通的 HTML 属性那样直接添加上去。

条件返回

`{}` 上面说了，JSX 可以放置任何表达式内容。所以也可以放 JSX，实际上，我们可以在 `render` 函数内部根据不同条件返回不同的 JSX。例如：

```
...
render () {
  const isGoodWord = true
  return (
    <div>
      <h1>
        React 小书
        {isGoodWord
          ? <strong> is good</strong>
          : <span> is not good</span>
        }
      </h1>
    </div>
  )
}
...
```

上面的代码中定义了一个 `isGoodWord` 变量为 `true`，下面有个用 `{}` 包含的表达式，根据 `isGoodWord` 的不同返回不同的 JSX 内容。现在页面上是显示 `React 小书 is good`。如果你把 `isGoodWord` 改成 `false` 然后再看页面上就会显示 `React 小书 is not good`。

如果你在表达式插入里面返回 `null` ，那么 React.js 会什么都不显示，相当于忽略了该表达式插入。结合条件返回的话，我们就做到显示或者隐藏某些元素：

```
...
render () {
  const isGoodWord = true
  return (
    <div>
      <h1>
        React 小书
        {isGoodWord
          ? <strong> is good</strong>
          : null
        }
      </h1>
    </div>
  )
}
...
```

这样就相当于在 `isGoodWord` 为 `true` 的时候显示 `<strong>is good</strong>`，否则就隐藏。

条件返回 JSX 的方式在 React.js 中很常见，组件的呈现方式随着数据的变化而不一样，你可以利用 JSX 这种灵活的方式随时组合构建不同的页面结构。

如果这里有些同学觉得比较难理解的话，可以回想一下，其实 JSX 就是 JavaScript 里面的对象，转换一下角度，把上面的内容翻译成 JavaScript 对象的形式，上面的代码就很好理解了。

JSX 元素变量

同样的，如果你能理解 JSX 元素就是 JavaScript 对象。那么你就可以联想到，JSX 元素其实可以像 JavaScript 对象那样自由地赋值给变量，或者作为函数参数传递、或者作为函数的返回值。

```
...
render () {
  const isGoodWord = true
  const goodWord = <strong> is good</strong>
  const badWord = <span> is not good</span>
  return (
    <div>
      <h1>
        React 小书
        {isGoodWord ? goodWord : badWord}
      </h1>
    </div>
  )
}
...
```

这里给把两个 JSX 元素赋值给了 `goodWord` 和 `badWord` 两个变量，然后把它们作为表达式插入的条件返回值。达到效果和上面的例子一样，随机返回不同的页面效果呈现。

再举一个例子：

```
...
renderGoodWord (goodWord, badWord) {
  const isGoodWord = true
  return isGoodWord ? goodWord : badWord
}

render () {
  return (
    <div>
      <h1>
        React 小书
        {this.renderGoodWord(
```

```
          <strong> is good</strong>,
          <span> is not good</span>
        )}
      </h1>
    </div>
  )
}
...
```

这里我们定义了一个 `renderGoodWord` 函数，这个函数接受两个 JSX 元素作为参数，并且随机返回其中一个。在 `render` 方法中，我们把上面例子的两个 JSX 元素传入 `renderGoodWord` 当中，通过表达式插入把该函数返回的 JSX 元素插入到页面上。

课后练习

- 用 React.js 构建未读消息组件
- JSX元素变量

> 因为第三方评论工具有问题，对本章节有任何疑问的朋友可以移步到 React.js 小书的论坛 发帖，我会回答大家的疑问。

下一节：组件的组合、嵌套和组件树

上一节：使用 JSX 描述 UI 信息