React.js 小书

<-- 返回首页

# ref 和 React.js 中的 DOM 操作


- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson21



在 React.js 当中你基本不需要和 DOM 直接打交道。React.js 提供了一系列的 `on*` 方法帮助我们进行事件监听，所以 React.js 当中不需要直接调用 `addEventListener` 的 DOM API；以前我们通过手动 DOM 操作进行页面更新（例如借助 jQuery），而在 React.js 当中可以直接通过 `setState` 的方式重新渲染组件，渲染的时候可以把新的 `props` 传递给子组件，从而达到页面更新的效果。

React.js 这种重新渲染的机制帮助我们免除了绝大部分的 DOM 更新操作，也让类似于 jQuery 这种以封装 DOM 操作为主的第三方的库从我们的开发工具链中删除。

但是 React.js 并不能完全满足所有 DOM 操作需求，有些时候我们还是需要和 DOM 打交道。比如说你想进入页面以后自动 focus 到某个输入框，你需要调用 `input.focus()` 的 DOM API，比如说你想动态获取某个 DOM 元素的尺寸来做后续的动画，等等。

React.js 当中提供了 `ref` 属性来帮助我们获取已经挂载的元素的 DOM 节点，你可以给某个 JSX 元素加上 `ref` 属性：

```
class AutoFocusInput extends Component {
  componentDidMount () {
    this.input.focus()
  }

  render () {
    return (
      <input ref={(input) => this.input = input} />
    )
  }
}

ReactDOM.render(
  <AutoFocusInput />,
  document.getElementById('root')
)
```

可以看到我们给 `input` 元素加了一个 `ref` 属性，这个属性值是一个函数。当 `input` 元素在页面上挂载完成以后，React.js 就会调用这个函数，并且把这个挂载

以后的 DOM 节点传给这个函数。在函数中我们把这个 DOM 元素设置为组件实例的一个属性，这样以后我们就可以通过 `this.input` 获取到这个 DOM 元素。

然后我们就可以在 `componentDidMount` 中使用这个 DOM 元素，并且调用 `this.input.focus()` 的 DOM API。整体就达到了页面加载完成就自动 focus 到输入框的功能（大家可以注意到我们用上了 `componentDidMount` 这个组件生命周期）。

我们可以给任意代表 HTML 元素标签加上 `ref` 从而获取到它 DOM 元素然后调用 DOM API。但是记住一个原则：**能不用 `ref` 就不用**。特别是要避免用 `ref` 来做 React.js 本来就可以帮助你做到的页面自动更新的操作和事件监听。多余的 DOM 操作其实是代码里面的“噪音”，不利于我们理解和维护。

顺带一提的是，其实可以给组件标签也加上 `ref` ，例如：

```
<Clock ref={(clock) => this.clock = clock} />
```

这样你获取到的是这个 `Clock` 组件在 React.js 内部初始化的实例。但这并不是什么常用的做法，而且也并不建议这么做，所以这里就简单提及，有兴趣的朋友可以自己学习探索。

## 课后练习

- 获取文本的长度

> 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：props.children 和容器类组件

上一节：更新阶段的组件生命周期