

React.js 小书

[<-- 返回首页](#)

事件监听

- 作者: [胡子大哈](#)
- 原文链接: <http://huziketang.com/books/react/lesson9>
- 转载请注明出处, 保留原文链接和作者信息。

在 React.js 里面监听事件是很容易的事情, 你只需要给需要监听事件的元素加上属性类似于 `onClick`、`onKeyDown` 这样的属性, 例如我们现在要给 `Title` 加上点击的事件监听:

```
class Title extends Component {
  handleClickOnTitle () {
    console.log('Click on title.')
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle}>React 小书</h1>
    )
  }
}
```

只需要给 `h1` 标签加上 `onClick` 的事件, `onClick` 紧跟着是一个表达式插入, 这个表达式返回一个 `Title` 自己的一个实例方法。当用户点击 `h1` 的时候, React.js 就会调用这个方法, 所以你在控制台就可以看到 `Click on title.` 打印出来。

在 React.js 不需要手动调用浏览器原生的 `addEventListener` 进行事件监听。React.js 帮我们封装好了一系列的 `on*` 的属性, 当你需要为某个元素监听某个事件的时候, 只需要简单地给它加上 `on*` 就可以了。而且你不需要考虑不同浏览器兼容性的问题, React.js 都帮我们封装好这些细节了。

React.js 封装了不同类型的事件, 这里就不一一列举, 有兴趣的同学可以参考官网文档: [SyntheticEvent - React](#), 多尝试不同的事件。另外要注意的是, 这些事件属性名都必须要用驼峰命名法。

没有经过特殊处理的话, 这些 `on*` 的事件监听只能用在普通的 HTML 的标签上, 而不能用在组件标签上。也就是说, `<Header onClick={...} />` 这样的写法不会有什么效果的。这一点要注意, 但是有办法可以做到这样的绑定, 以后我们会提及。现在只要记住一点就可以了: 这些 `on*` 的事件监听只能用在普通的 HTML 的标签上, 而不能用在组件标签上。

event 对象

和普通浏览器一样，事件监听函数会被自动传入一个 `event` 对象，这个对象和普通的浏览器 `event` 对象所包含的方法和属性都基本一致。不同的是 `React.js` 中的 `event` 对象并不是浏览器提供的，而是它自己内部所构建的。`React.js` 将浏览器原生的 `event` 对象封装了一下，对外提供统一的 API 和属性，这样你就不用考虑不同浏览器的兼容性问题。这个 `event` 对象是符合 W3C 标准（[W3C UI Events](https://www.w3.org/TR/2003/WD-DOM-Level-3-Events-20030331/)）的，它具有类似于 `event.stopPropagation`、`event.preventDefault` 这种常用的方法。

我们来尝试一下，这次尝试当用户点击 `h1` 的时候，把 `h1` 的 `innerHTML` 打印出来：

```
class Title extends Component {
  handleClickOnTitle (e) {
    console.log(e.target.innerHTML)
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle}>React 小书</h1>
    )
  }
}
```

再看看控制台，每次点击的时候就会打印“React 小书”。

关于事件中的 `this`

一般在某个类的实例方法里面的 `this` 指的是这个实例本身。但是你在上面的 `handleClickOnTitle` 中把 `this` 打印出来，你会看到 `this` 是 `null` 或者 `undefined`。

```
...
  handleClickOnTitle (e) {
    console.log(this) // => null or undefined
  }
...
```

这是因为 `React.js` 调用你所传给它的方法的时候，并不是通过对象方法的方式调用（`this.handleClickOnTitle`），而是直接通过函数调用（`handleClickOnTitle`），所以事件监听函数内并不能通过 `this` 获取到实例。

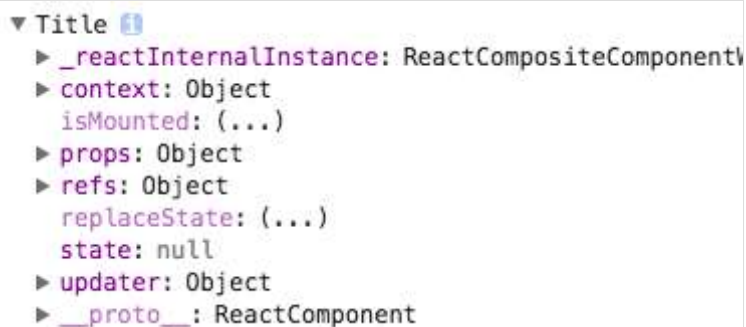
如果你想在事件函数当中使用当前的实例，你需要手动地将实例方法 `bind` 到当前实例上再传入给 `React.js`。

```
class Title extends Component {
  handleClickOnTitle (e) {
```

```
    console.log(this)
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle.bind(this)}>React 小书</h1>
    )
  }
}
```

`bind` 会把实例方法绑定到当前实例上，然后我们再把绑定后的函数传给 `React.js` 的 `onClick` 事件监听。这时候你再看看，点击 `h1` 的时候，就会把当前的实例打印出来：



```
▼ Title
  ▶ _reactInternalInstance: ReactCompositeComponent
  ▶ context: Object
    isMounted: (...)
  ▶ props: Object
    onClick: function
  ▶ refs: Object
  ▶ replaceState: (...)
  ▶ state: null
  ▶ updater: Object
  ▶ __proto__: ReactComponent
```

你也可以在 `bind` 的时候给事件监听函数传入一些参数：

```
class Title extends Component {
  handleClickOnTitle (word, e) {
    console.log(this, word)
  }

  render () {
    return (
      <h1 onClick={this.handleClickOnTitle.bind(this, 'Hello')}>React 小书</h1>
    )
  }
}
```

这种 `bind` 模式在 `React.js` 的事件监听当中非常常见，`bind` 不仅可以帮我们帮事件监听方法中的 `this` 绑定到当前组件实例上；还可以帮助我们在渲染列表元素的时候，把列表元素传入事件监听函数当中——这个将在以后的章节提及。

如果有些同学对 `JavaScript` 的 `this` 模式或者 `bind` 函数的使用方式不是特别了解到话，可能会对这部分内容会有些迷惑，可以补充对 `JavaScript` 的 [this](#) 和 [bind](#) 相关的知识再来回顾这部分内容。

总结

为 `React` 的组件添加事件监听是很简单的事情，你只需要使用 `React.js` 提供了一系列的 `on*` 方法即可。

React.js 会给每个事件监听传入一个 `event` 对象，这个对象提供的功能和浏览器提供的功能一致，而且它是兼容所有浏览器的。

React.js 的事件监听方法需要手动 `bind` 到当前实例，这种模式在 React.js 中非常常用。

课后练习

- [不能摸的狗（一）](#)

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 `state` 和 `setState`

上一节：组件的组合、嵌套和组件树

如果你觉得小书写得还不错，可以请胡子大哈喝杯茶 :)

赞赏

或者传播一下知识也是一个很好的选择

6 条评论，6 人参与。



我有话说...

使用社交帐号登录

发布前先点击左边的按钮登录



最新评论



木辛1 丕申 • 6月28日 22:52

无论是 `onClick={this.handleClickOnTitle.bind(this)}` 或是 `onClick={e => this.handleClickOnTitle()}` 都不符合最佳实践。前者反复 `bind` 浪费性能，后者每次都创建新的 `callback`。因此最佳做法还是在 `constructor` 中绑定 `this.handleClickOnTitle = this.handleClickOnTitle.bind(this)`。

顶 • 回复 • 分享»



夜那么白 • 4月25日 17:32

新的es好像支持这种模式？`handleClickOnTitle = (e) => console.log(this)`，这个`this`好像会自动绑定`Title`。

顶 • 回复 • 分享»



_Rhythhm • 4月25日 15:56

`handleClickOnTitle (e) { console.log(this) // => null or undefined }` 这里为什么是`null`或`undefined`，我试了都是`null`，想知道为什么是`null`呢？

顶 • 回复 • 分享»



恁月 • 4月21日 11:16

小白有问题，望见笑。`<h1 onClick={this.handleClickOnTitle}>React 小书</h1>`这一句，`onClick`方法调用事件监听函数，`handleClickOnTitle`后面为啥没有括号

顶 • 回复 • 分享»



木心 恁月 • 6月19日 14:29

如果你加括号的话，那是不是意味着调用了一次函数？这个函数可能会 `return` 出一个值，那也就意味着，你绑定的是 `return` 出的这个值了，而不是函数

顶 • 回复 • 分享»



卑鄙的我的Baby 恁月 • 5月3日 17:11

在js中函数是一个对象。 `this.handleClickOnTitle` 表示函数本身。`this.handleClickOnTitle()` 意味着函数的一次调用。

顶 • 回复 • 分享»

友言？