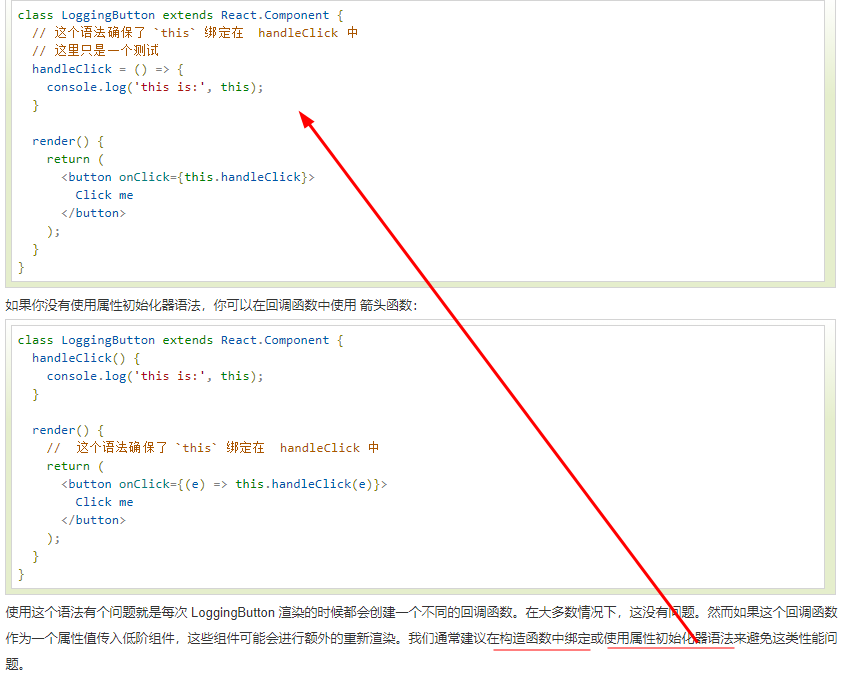
如果你没有在方法后面添加 () ，例如 onClick={this.handleClick}，你应该为这个方法绑定 this。



constructor this.state bind(this)





通过箭头函数bind(this)



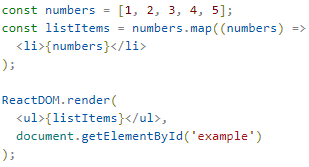
onClick时传递参数



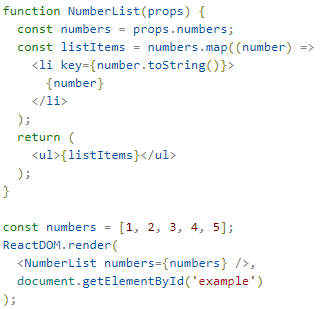
通过bind的方式传递参数



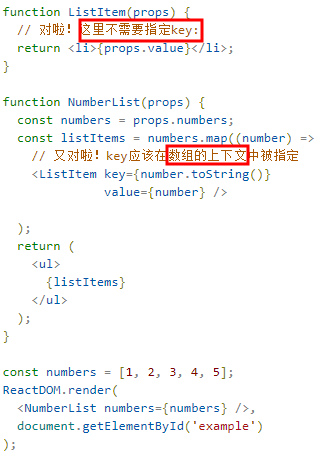
一旦showWarning变化，WarningBanner就会重新渲染



变量存放着数组，数组存放的元素为dom元素



动态渲染列表



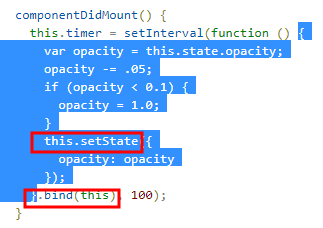
在调用组件的时候给key



数组元素中使用的 key 在其兄弟之间应该是独一无二的。然而，它们不需要是全局唯一的。当我们生成两个不同的数组时，我们可以使用相同的键。



组件/类的生命周期



bind(this)



abort()终止ajax



onChange监听input的变化



将事件传入子组件



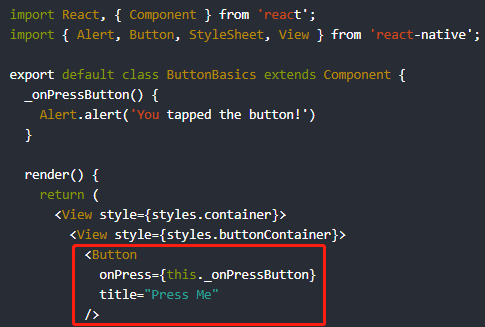
下拉列表



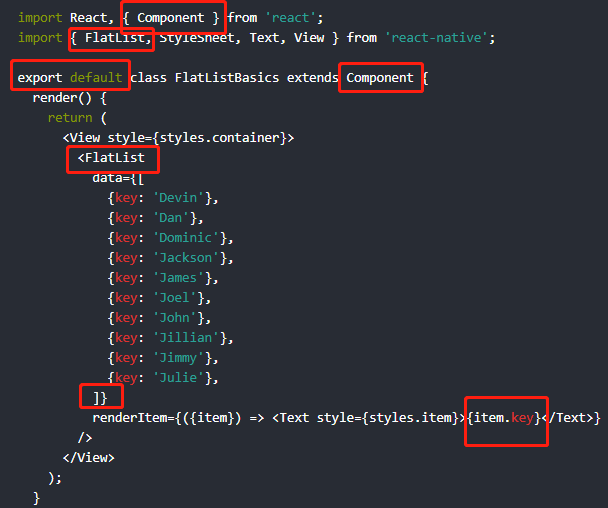
在onChange中获取dom。给变量setState要用中括号



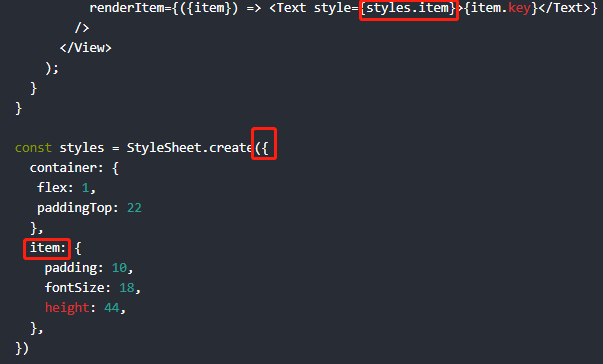
refs。可以在调用的时候bind(this)



button的onPress事件



flatList列表



style的话括号中传入一个对象



获取路径中的hash值

window.location.hash

window.onhashchange

C:\Users\林\AppData\Local\Temp\WeChat Files\bf2e320893c25e3f0673f5d6d294eb9.png

history前端路由每次都向服务器发送请求

react组件：

1. 函数式定义的无状态组件
   1. 性能更高效、代码更简洁
   2. 不需要管理/维护 组件的生命周期
   3. 没有 this。在Stateless中 this 是 undefined 。

绑定方法的四种方式：

//第一种方法:构造函数中绑定

this.onClickList = this.onClickList.bind(this)

//第二种方法: 在render()行内事件使用bind来绑定

return <div onClick={this.onClickList.bind(this)}>点我点我点我</div>

//第三种方法: 使用箭头函数 =>

onClickList = () => {}

//第四种，当然，你要在render()行内使用箭头函数也行

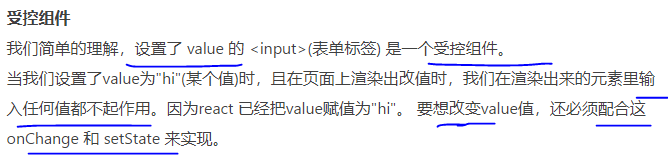
<div onClick={() => this.onClickList()}>点我点我点我</div>

**箭头函数**的方法，代码量比第一种少

官方说在 render 中创建函数(第二，和第四种)可能会有性能问题。但往往需要**传递参数**或者**回调**时，都得用到。

jsx优点:

执行更快，因为它在编译为 JavaScript 代码后进行了优化



受控组件



通过函数setState

**子组件向父组件通信**

通过 props 传递一个函数，子组件调用这个函数，并将子组件需要传递的数据作为参数，传递给父组件。

状态管理工具 mobx redux

路由：

<Route exact path="/detail/:id" component={Detail}/>//传参

使用this.props.match.params获取url传过来的参数

隐式传参：

<button onClick={() => this.props.history.push({

pathname: '/detail',

state: {

id: 3

}

})}>通过函数跳转</button>

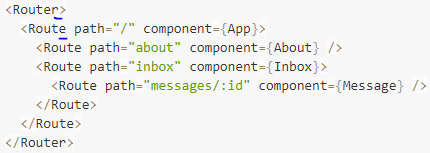
路由传参

就可以使用this.props.history.location.state获取home传过来的参数

路由二：

import { Router, Route, Link } from 'react-router'

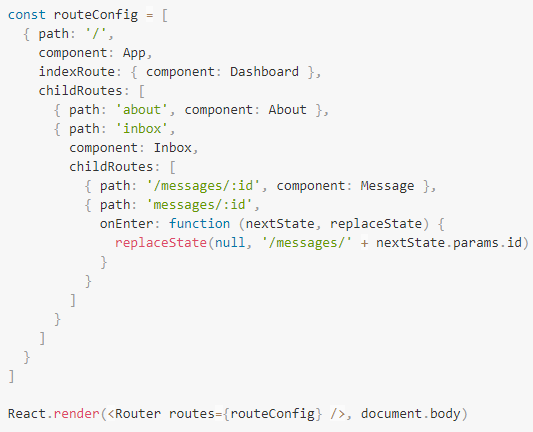
<Link to="/about">About</Link>



路由嵌套，路由传参



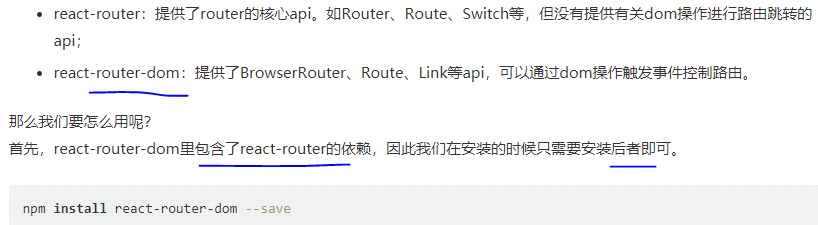
Redirect重定向

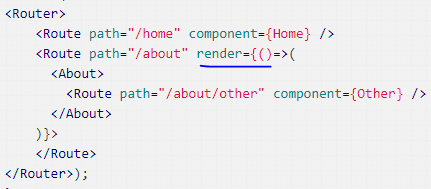


路由的抽离

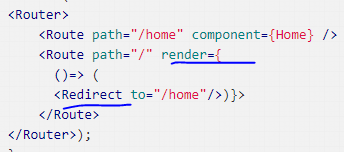


括号中在匹配中可以要也可以不要匹配，路由传参



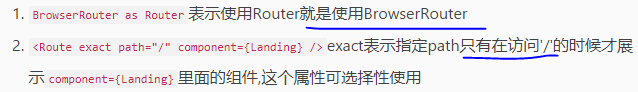


路由嵌套



Route标签中的html

路由重定向



BrowserRouter

<NavLink exact to='/'>Jsdemoa</NavLink>

<NavLink to='/Jsdemob' activeClassName='active'>Jsdemob</NavLink>