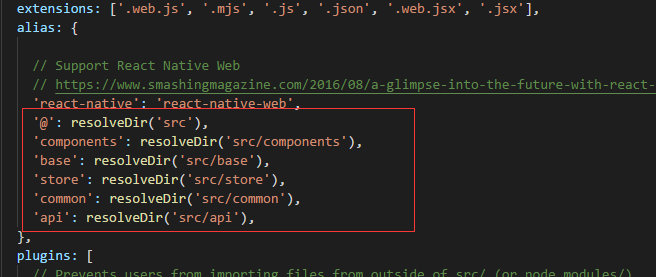
create-react-app生成的项目文，看不到webpack相关的配置文件

npm **run** eject 可以暴露出来配置文件

修改配置

config文件夹下webpack.config.dev.js

**路径别名配置**



**Stylus配置**

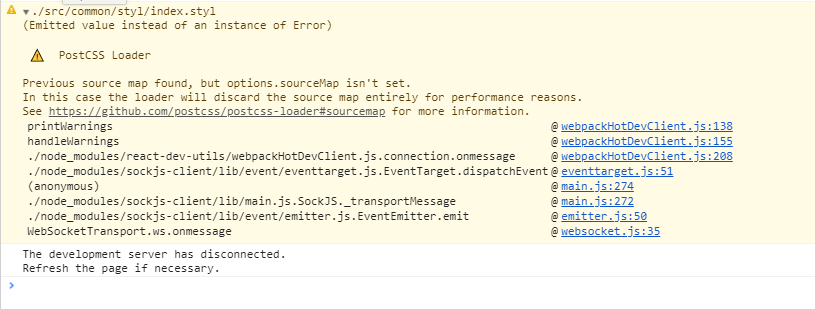
安装stylus stylus-loader

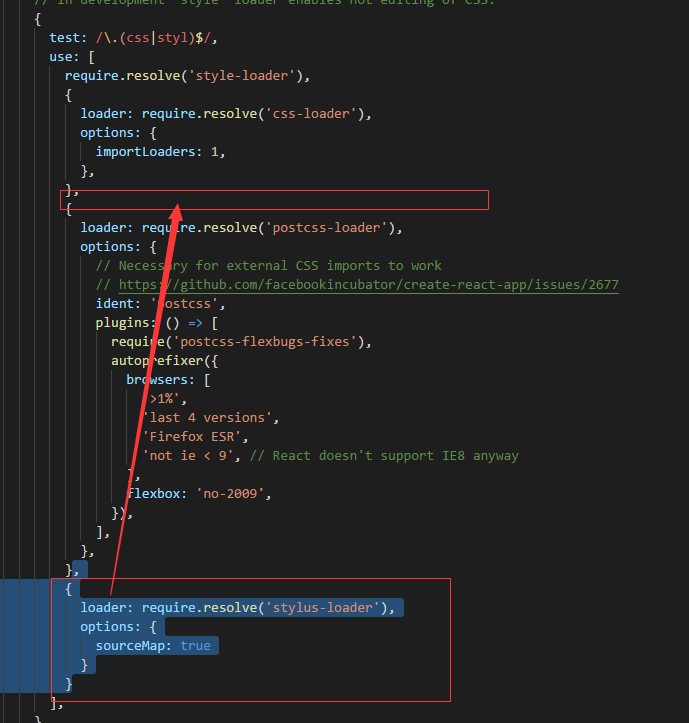
Emitted value instead of an instance of Error)

⚠️ PostCSS Loader

Previous source map found, but options.sourceMap isn't set.

In this case the loader will discard the source map entirely for performance reasons.



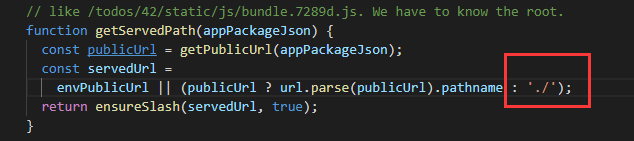




npm run build 打包出来的项目放到非根目录访问不了

方法一下1.想办法弄到根目录

方法二：1.eject后修改config文件夹下publicUrl路径 ‘/’改为 ‘./’



1. BrowserRouter改用hashRouter

import { HashRouter as Router, Route, Switch,Redirect} from 'react-router-dom'

或者直接import { Router}

1.初始化数据

Vue中data函数中初始化数据以及类型

data() {

return {

dots: [],

currentPageIndex: 0

}

}

React中

//在构造函数中对状态赋初始值

constructor(props) {

super(props);

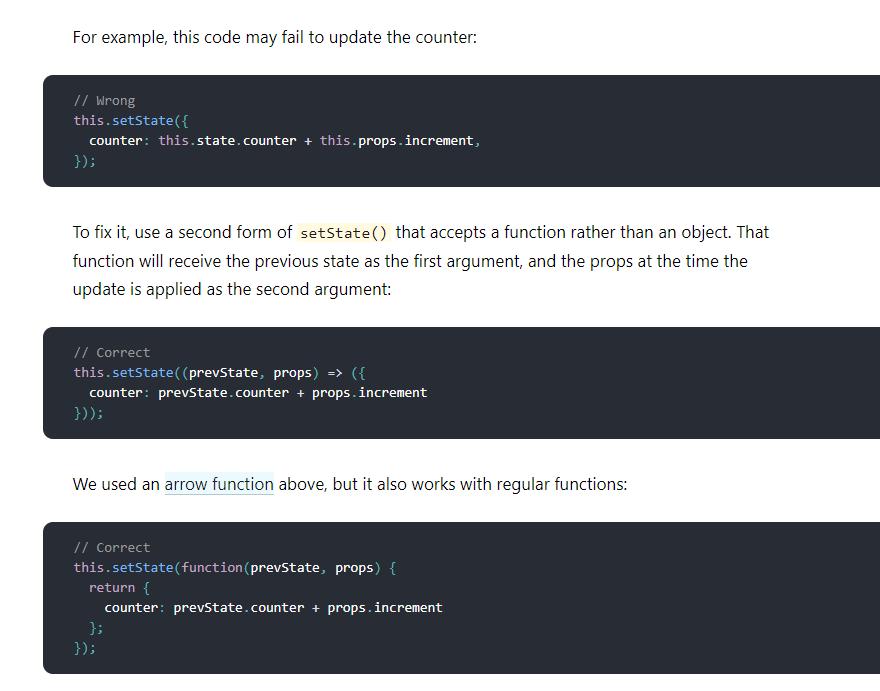
this.state = {dots: []};

}

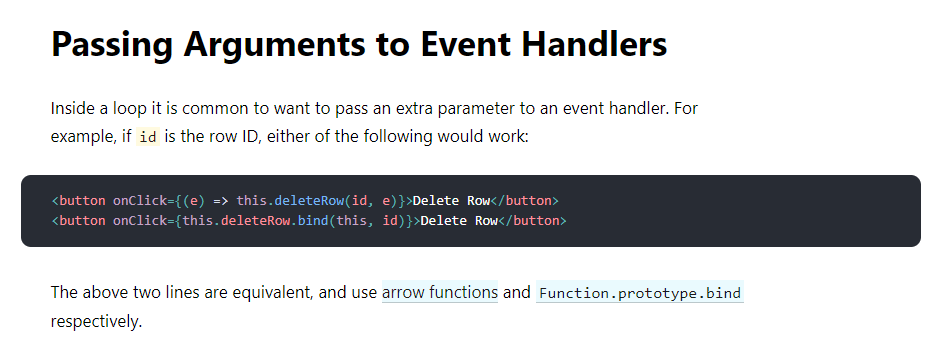
The only place where you can assign this.state is the constructor.

<https://reactjs.org/docs/state-and-lifecycle.html>

2.

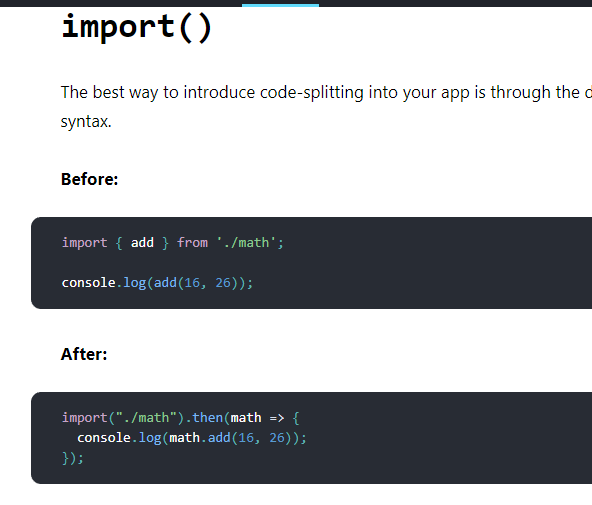


**3.[Handling Events](https://reactjs.org/docs/handling-events.html)**



We know that [props are read-only](https://reactjs.org/docs/components-and-props.html" \l "props-are-read-only).

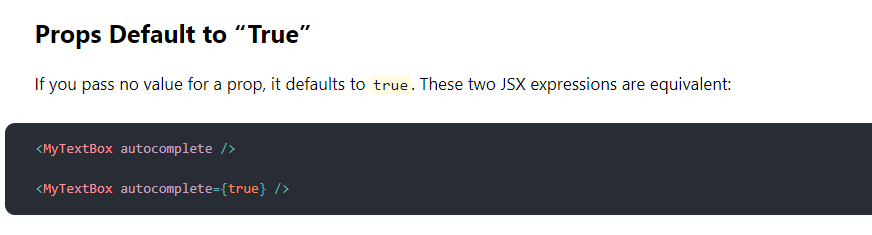
4.



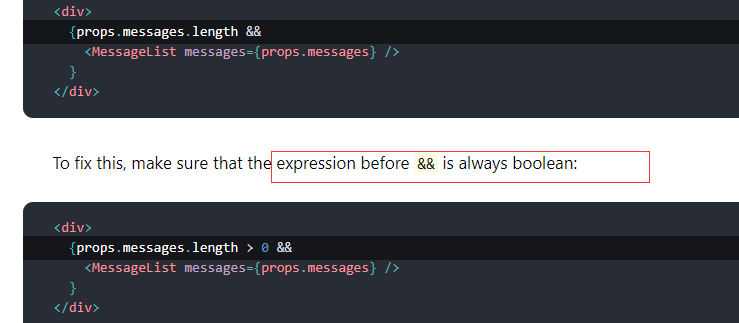
When Webpack comes across this syntax, it automatically starts code-splitting your app.

5. ****a higher-order component is a function that takes a component and returns a new component.****

### 6.Props Default to “True”



7.条件渲染



1. 动态绑定class style

Vue

<div class="buttonBox" @click="login" :class="{disable:btnDisable===true}">

react

<div className={ 'eye ' + (this.state.isOpen ? 'open' : '')} onClick={this.switchEye}></div>

style={{display: this.isSelf ? 'none' : 'block'}}

1. 根据输入框情况计算按钮是否可点击

Vue

放在watch中去观测data下的一个属性btnDisable

data: () => ({

btnDisable

})

watch中根据情况去修改 this.btnDisable

React中放在shouldComponentUpdate 观测constructor定义的一个全局属性，注意不能是this.state中定义。。因为这样会陷入死循环

constructor() {

super()

this.state = {

mobile: '',

password: '',

}

this.btnDisable = false

}

shouldComponentUpdate中根据情况去修改 this.btnDisable ,而不是this.state.btnDisable

shouldComponentUpdate

Return bool 判断组件是否应该重新渲染，默认是true

10 react-router4 不再推荐将所有路由规则放在同一个地方集中式路由，子路由应该由父组件动态配置，组件在哪里匹配就在哪里渲染，更加灵活

1. 如何想根据一个变量去控制另一个变量（按钮）的状态，vue中有watch很容做的，

React中需要用shouldComponentUpdate

并且需要依赖的变量需要定义在state中，才可以被参数nextState观测到，被改变的变量

不能是state中定义的数据，可以使用this.xx作为一个组件中的全局变量定义并使用

原因是如果作为计算结果的变量定义在state中的话会陷入死循环

总结：作为依赖的计算条件的变量放在state中，计算结果的变量定义为全局的

全局的不会自动更新dom，state下的才会

React不监听数据对象，而是通过手动调用setState()方法来触发virtueDiff，以此对比前后来个状态的不同，然后针对性的更改变化了的DOM结构实现数据更新。这是我的理解，欢迎大神拍砖

1. 传递参数的函数 需要使用监听函数的形式 直接方法传递无效

Vue @click="getCode(mobile)"

React onClick={(e) => this.getCode(this.state.mobile)}

# React的onClick事件，加载页面的时候会自动执行，执行过后，onClick事件就失效，再点击就没有作用 --->传参函数使用箭头函数就行了

插槽

Vue <slot></slot>

React {this.props.children}

12.

动态绑定多个class

Vue

<span :class="[{'active':index<=step+(step-1)},{'setp-line':index%2===1}]" v-for="(item,index) in stepLine"></span>

React 借助classNames库

{

this.state.stepLine.map((item,index) => {

return <span className={classNames(index<=step+(step-1)? 'active' : '', (index%2===1 ? 'setp-line' : ''))} key={index}></span>

})

}

子组件接受父组件的传递值

Vue : props

React static defaultProps = { loop: true}

如果需要加类型推断引入

import PropTypes from 'prop-types'

static propTypes = {loop: PropTypes.bool}

子组件需要传递值到父组件中，

Vue emit 派发事件并把值作为参数派发过去--->父组件监听事件并取到值

React 父组件把方法作为属性 传递给子组件---->子组件通过this.props.xx直接使用父组件的方法并把值传递进去

子

// this.$emit('selectedValue', this.selectValue)

this.props.selectedCarValue(this.selectValue)

父

selectedCarValue(val) { //选择车牌号

this.setState({carno: val})

}

<CarSelectNo ref="selectCarNo" defaultValue={this.state.carno} selectedCarValue={this.selectedCarValue}></CarSelectNo>

获取元素或组件的引用

Vue <div ref=’sliderDom’></div>

this.$refs.sliderDom ----是一个dom

React

constructor(props) {this.sliderDom = React.createRef() }

<div className="slider" ref={this.sliderDom}>

this.sliderDom.current -----是一个dom

或者直接加字符串 <div className="slider" ref=’sliderDom’>

### this.refs.sliderDom 该元素的引用 旧版 API

如果是组件

this.confirm = React.createRef()

<Confirm text={this.state.tipsText} confirmType='2' ref="confirm"></Confirm>

this.refs.confirm.show()

一下2中写法都可以

// this.refs.scroll.scrollTo(0, 0, 500) //ref="scroll"

this.scroll.current.scrollTo(0, 0, 500) //ref={this.scroll}

img元素上的本地图片引用

Vue <img src="../../common/image/banner2.png" alt=""/>

React <img src={require('../../common/image/banner.png')} alt="" />

<img alt='' className='pic' src={item.get('imgUrl')} />

**路由跳转**

1.路由标签方式跳转

Vue

<router-link tag="div" class="card-item" to=""></router-link>

react

import {NavLink} from 'react-router-dom'

<NavLink to="/login" className="card-item"></NavLink>

2.js方式跳转

Vue this.$router.push('/login')

React

import {withRouter} from "react-router-dom";

this.props.history.push("/login");

export default withRouter(Home)

路由传递参数

1.路由rougte上配置

在子组件中使用Context，我们可以获得router对象，如下面例子~

当然，这种方法慎用~尽量不用。因为react不推荐使用contex哦。在未来版本中有可能被抛弃哦。

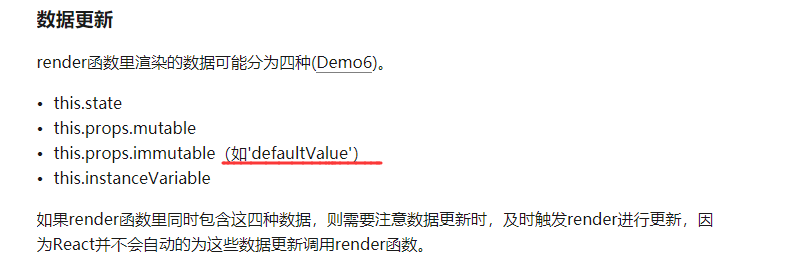
**其他**

1. 如果不想使用bind保存this到函数上 可以把方法改为箭头函数

事件点击就报错 并且this的值还为undefined 可能就是方法没有绑定this导致

Input 的默认值defaultValue

问题如果input的默认值是通过给绑定defaultValue给的，那么通过异步再次更新数据的时候，即使数据更新了，默认值的显示也不会发生更改 改成 value就可以了



react中子组件如何时时监听父组件传递过来的数据

shouldComponentUpdate

中去做

父组件传递

<CarSelectNo ref="selectCarNo" defaultValue={this.state.carno} selectedCarValue={this.selectedCarValue}></CarSelectNo>

跟vue中一样，该监听事件继续监听事件。。只是子组件中派发事件变成了调用事件

子组件中监听数据变化

shouldComponentUpdate (nextProps, nextState) {

console.log(nextProps);

this.selectValue = nextProps.defaultValue

return true

}

widow.innerHeight



Constructor() {

this.onChange = this.onChange.bind(this) //构造函数中设置

}

onChange () {

   this.setState({ query: event.target.value });

 }

------>

onChange = event => {

   this.setState({ query: event.target.value });

 }

1. 通过使用三元运算符可以将函数体缩短为一行代码。因此可以省略函数体和 return 语句。

React 只提供了一个细小的 API 表面区域，因此开发人员必须习惯于 JavaScript 提供的所有功能。这句话并非没有任何理由：“成为 React 开发者也会让你成为更好的 JavaScript 开发者”。让我们通过重构一个高阶组件来回顾一下学到的 JavaScript 的一些方面

React 的视野更为广阔

<https://mp.weixin.qq.com/s/VZxUFvcHY4lpdmvA1a8dyQ>

与渐进式 JavaScript 的不同之处在于，React 采用新工具时不必重构大量的代码

与 Ember 或者 Angular 这样的整体型框架不同，React 具有足够的灵活性来应对不可避免的变化

# **非路由组件如何使用this.props.history**

正常情况下 只有Router 的component组件能够自动带有三个属性 如下的Home 组件有

如果是路由组件，访问history一般都是通过this.props.history来操作history

* 而非路由组件通过 {withRouter} 来使用this.props.history

<https://www.jianshu.com/p/1a6e31a7e923>

import React from "react"; import {withRouter} from "react-router-dom"; // 引入withRouter class MyComponent extends React.Component { ... myFunction() { this.props.history.push("/some/Path"); } ... } export default withRouter(MyComponent); // 用withRouter包裹该class ---------------------------------------------------------------------------- this.props.location.pathname 获取当前的url

## immutable取值

多层级嵌套的对象取值

getIn([‘层级1’，‘层级2’])

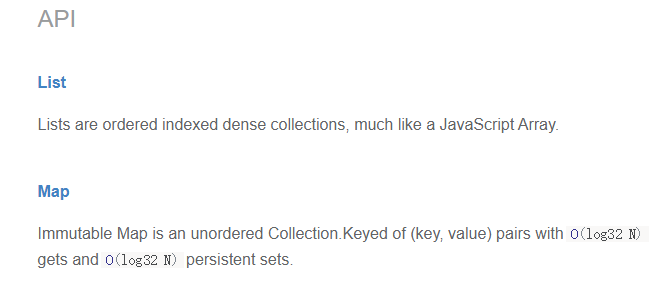
一层级取值

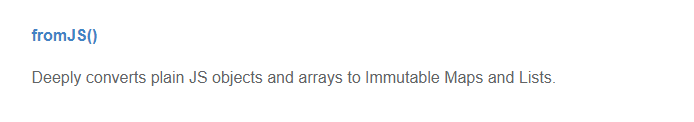
get(‘key’）

Immutable对象 可以用toObject转为原生js对象

Immutable数组 可以用toJs转为原生js数组

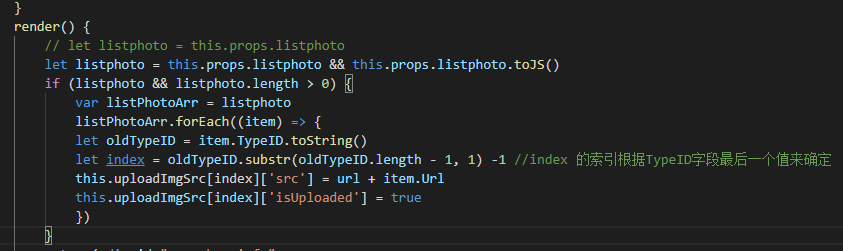
<https://facebook.github.io/immutable-js/docs/#/>





<https://segmentfault.com/a/1190000012156192>

如果想要redux获取过来的数据自动更新到页面上，替换数据的操作放在render里



iT营学习

1.搭建React的开发环境的第二种方法（新-未来推荐）

npx create-react-app my-app

npx create-react-app reactdemo这条命令会临时安装 create-react-app 包，命令完成后create-react-app 会删掉，不会出现在 global 中。下次再执行，还是会重新临时安装。

npx 会帮你执行依赖包里的二进制文件。

再比如 npx http-server 可以一句话帮你开启一个静态服务器

2.Es6中的super可以用在类的继承中，super关键字，它指代父类的实例（即父类的this对象）。子类必须在constructor方法中调用super方法，否则新建实例时会报错。这是因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工。如果不调用super方法，子类就得不到this对象。

为什么官方的列子里面写个super(props)：

只有一个理由需要传递props作为super()的参数，那就是你需要在构造函数内使用this.props

那官方提供学习的例子中都是写成super(props)，所以说写成super(props)是完全没问题的，也建议就直接这样写。

1. 组件名称首字母大写、组件类名称首字母大写

4.在以类继承的方式定义的组件中，为了能方便地调用当前组件的其他成员方法或属性（如：this.state），通常需要将事件处理函数运行时的 this 指向当前组件实例。

绑定事件处理函数this的几种方法：

第一种方法：

run(){

alert(this.state.name)

}

<button onClick={this.run.bind(this)}>按钮</button>

第二种方法：

构造函数中改变

this.run = this.run.bind(this);

run(){

alert(this.state.name)

}

<button onClick={this.run>按钮</button>

第三种方法：

run=()=> {

alert(this.state.name)

}

<button onClick={this.run>按钮</button>

图片资源引入的3种方式

1. import logo from '../assets/images/1.jpg';

<img src={logo} />

2. <img src={require('../assets/images/1.jpg')} />

3. <img src="https://www.baidu.com/img/xinshouye\_353af22a7f305e1fb6cfa259394dea9b.png" />

//动态路由的传值

to={`/content/${value.aid}`}>{value.title}</Link>

//获取动态路由的传值

//Content.js

console.log(this.props.match.params.aid);

<Link to={`/productcontent?aid=${value.aid}`}>{value.title}</Link>

console.log(url.parse(this.props.location.search,true));

var query=url.parse(this.props.location.search,true).query;

/\*

约束性和非约束性组件:

非约束性组:<input type="text" defaultValue="a" /> 这个 defaultValue 其实就是原生DOM中的 value 属性。

这样写出的来的组件，其value值就是用户输入的内容，React完全不管理输入的过程。

约束性组件：<input value={this.state.username} type="text" onChange={this.handleUsername} />

这里，value属性不再是一个写死的值，他是 this.state.username, this.state.username 是由 this.handleChange 负责管理的。

这个时候实际上 input 的 value 根本不是用户输入的内容。而是onChange 事件触发之后，由于 this.setState 导致了一次重新渲染。不过React会优化这个渲染过程。看上去有点类似双休数据绑定

\*/



React生命周期函数

组件加载的时候触发的函数：

constructor 、componentWillMount、 render 、componentDidMount

组件数据更新的时候触发的生命周期函数：

shouldComponentUpdate、componentWillUpdate、render、componentDidUpdate

你在父组件里面改变props传值的时候触发的：

componentWillReceiveProps

组件销毁的时候触发的：

componentWillUnmount

必须记住的生命周期函数：

\*加载的时候：componentWillMount、 render 、componentDidMount（dom操作）

更新的时候：componentWillUpdate、render、componentDidUpdate

\*销毁的时候： componentWillUnmount

高阶组件

1. 定义

export default (WrappedComponent) => {

return class extends Component {

constructor(props) {

super(props)

this.state = {

codeBtnText: '发送验证码'

}

this.getCode = this.getCode.bind(this)

this.postVcode = this.postVcode.bind(this)

}

componentWillMount() {

}

render() {

return (

<div>

<WrappedComponent getCode={this.getCode} state={this.state} {...this.props}/>

</div>

)

}

}

}

高阶组件内部定义的方法或者数据需要被其他组件使用时需要作为属性传递出去，如getCode方法与state中的按钮文本属性

1. 其他组件中使用

把当前组件传递到高阶组件中

import HOC from 'common/js/HOC'

export default withRouter(HOC(Forgetpassword))

或者直接使用装饰器模式---记得配置package babel下plugin

import HOC from 'common/js/HOC'

@HOC

class Forgetpassword extends Component{

render中由以前的

onClick={()=>this.getCode(this.state.mobile)}>{this.state.codeBtnText}</div>

------>改为接受props中的方法与属性状态

onClick={()=>this.props.getCode(this.state.mobile)}>{this.props.state.codeBtnText}</div>

package装饰器解析配置

"babel": {

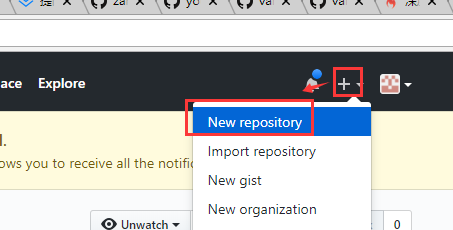
"presets": [

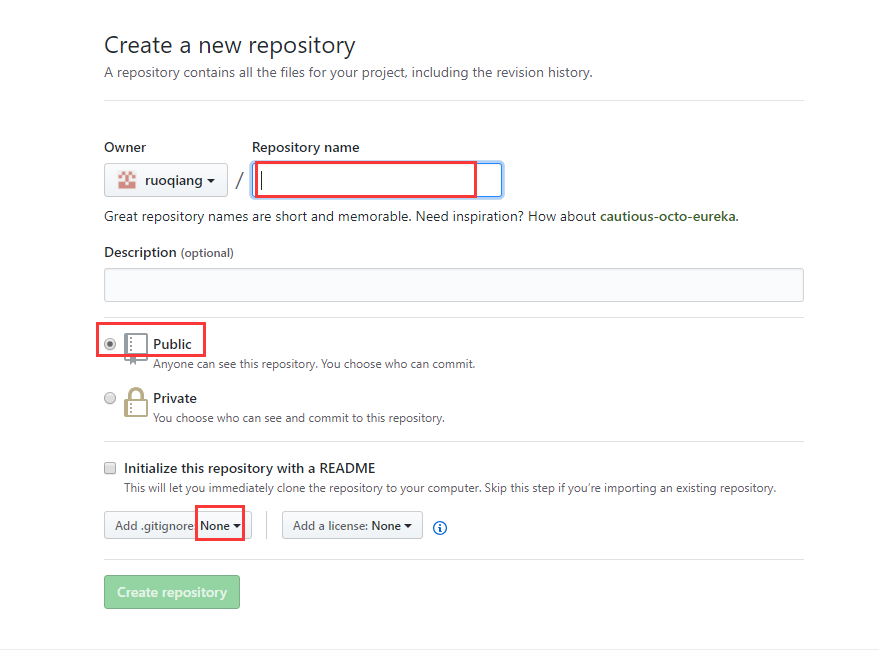
"react-app"

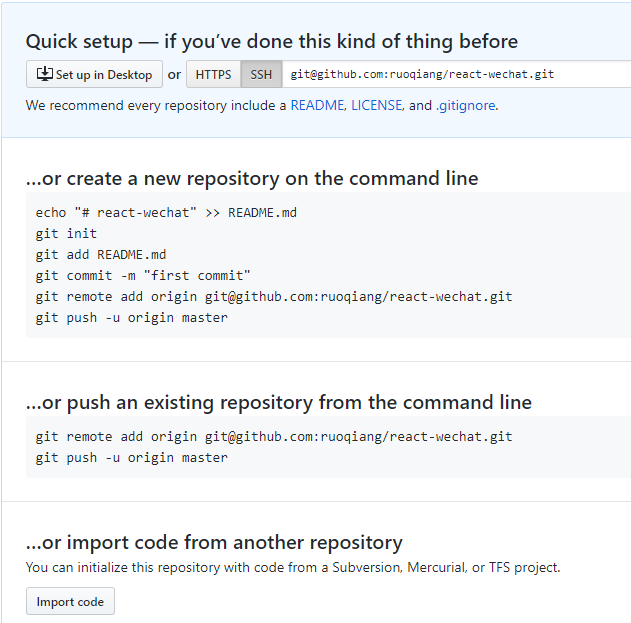
],

"plugins": ["transform-decorators-legacy"]

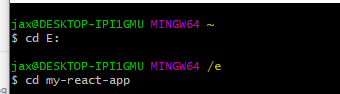
},







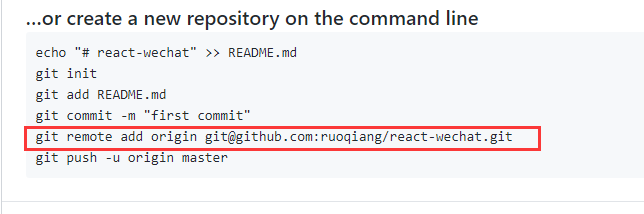
打开Git bash进入项目



git init

git add .

git commit –m "first commit"

按照提示执行

git remote add origin git@github.com:ruoqiang/react-wechat.git

git push -u origin master