**React入门学习笔记**

# 搭建环境：

准备：

* 必须安装nodejs 注意：安装nodejs稳定版本
* 安装cnpm用cnpm替代npm

npm install -g cnpm --registry=https://registry.npm.taobao.org

* 用yarn替代npm

cnpm install -g yarn 或者 npm install -g yarn

搭建：

1. 方法一

<https://reactjs.org/docs/create-a-new-react-app.html>

必须要安装nodejs 注意：安装nodejs稳定版本

安装脚手架工具 （单文件组件项目生成工具） 只需要安装一次

npm install -g create-react-app / cnpm install -g create-react-app

创建项目 （可能创建多次） 找到项目要创建的目录：create-react-app reactdemo

cd reactdemo

npm start yarn start运行项目

npm run build yarn build 生成项目

1. 方法二

<https://reactjs.org/docs/create-a-new-react-app.html>

npx create-react-app reactdemo

……

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。

# 03

1、所有的模板要被一个根节点包含起来

2、模板元素不要加引号

3、{}绑定数据

4、绑定属性注意：

class 要变成 className

for 要变成 htmlFor

style属性和以前的写法有些不一样

<div style={{'color':'blue'}}>{this.state.title}</div>

<div style={{'color':this.state.color}}>{this.state.title}</div>

style最外层的大括号表示要给style绑定属性，内层的括号表示绑定的属性是一个对象形式

引入图片：

本地图片：import logo from './assets/logo.png' <img src={logo}/>

远程图片：直接这样写<img src="http://xxx/xxx.png"/>

5、循环数据要加key

6、组件的构造函数中一定要注意 super

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

constructor(props){

super(props); /\*用于父子组件传值 固定写法\*/

this.state={

userinfo:'张三'

}

}

7、组件名称首字母大写、组件类名称首字母大写

# 04

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>

注意这里要用bind，因为xxx.bind(this)会返回一个函数，但是不会执行；

但是xxx.call(this) 就会执行。

* 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>

执行方法传值：

<button onClick={ this.setName.bind(this, '张三','李四') }>按钮</button>

# 05

* Event.target

<button aid="123" onClick={this.run}>事件对象</button>

run=(event)=>{

// console.log(event);

// alert(event.target); /\*获取执行事件的dom节点\*/

event.target.style.background='red';

//获取dom的属性

alert(event.target.getAttribute('aid'));

this.setState({

username:e.target.value

})

}

* Ref

获取dom节点

1、给元素定义ref属性 <input ref="username" />

2、通过this.refs.username 获取dom节点

<input ref="username" onChange={this.inputChange}/>

inputChange=()=>{

let val=this.refs.username.value;

this.setState({

username:val

})

}

* 双向数据绑定

model改变影响View view改变反过来影响model

react是单向的，即model改变影响View；

但是通过对改变view的事件的监听，在事件里通过setState改变model，从而模拟view影响model;

# 06

* 非约束性组件：

<input type=”text” defaultValue=”a” />

这个defaultValue其实就是原生DOM中的value属性。这样写出来的组件，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 type=”text” value={this.state.username} 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.handleUsername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。

# 10

React中的组件: 解决html 标签构建应用的不足。

使用组件的好处：把公共的功能单独抽离成一个文件作为一个组件，哪里使用哪里引入。

父子组件：组件的相互调用中，我们把调用者称为父组件，被调用者称为子组件

* 父子组件传值：

父组件给子组件传值：

1.在调用子组件的时候定义一个属性 <Header msg='首页'></Header>

2.子组件里面 this.props.msg

说明：父组件不仅可以给子组件传值，还可以给子组件传方法,以及把整个父组件传给子组件。msg={this}

父组件主动获取子组件的数据:

1、调用子组件的时候指定ref的值 <Header ref='header'></Header>

2、通过this.refs.header 获取整个子组件实例

注意：父组件获取子组件实例的时候，要在子组件加载完成后获取。

# 11

父组件给子组件传值：

defaultProps: 父组件传值中，如果父组件调用子组件的时候不给子组件传值，可以在子组件中使用defaultProps定义的默认值。

Header.defaultProps={

title:'标题'

}

propTypes:验证父组件传值的类型合法性

1. 引入import PropTypes from ‘prop-types’
2. Header.propTypes = {

name: PropTypes.string

}

都是定义在子组件中。

# 12

当需要引入第三方插件时，可以这样学习：百度搜索npmjs->在搜索栏中搜索需要引入的插件->看官方文档

--save表示把该模块写入到配置文件 package.json

React获取接口的数据：

React中没有提供专门的请求数据的模块，但是我们可以使用任何第三方请求数据模块请求数据。

* axios

1.安装axios模块 npm install axios –-save/ cnpm install axios –-save

2.在哪里使用就在哪里引入import axios from ‘axios‘

3.看文档使用

var api='http://www.phonegap100.com/appapi.php?a=getPortalList&catid=20'; //接口后台允许了跨域

axios.get(api)

.then((response) => {

this.setState({

list:response.data.result

});

})

.catch((error) => {

console.log(error);

});

* fetch-jsonp

1、安装 npm install fetch-jsonp --save

2、import fetchJsonp from 'fetch-jsonp'

3、看文档使用

fetchJsonp('/users.jsonp')

.then(function(response) {

return response.json()

}).then(function(json) {

console.log('parsed json', json)

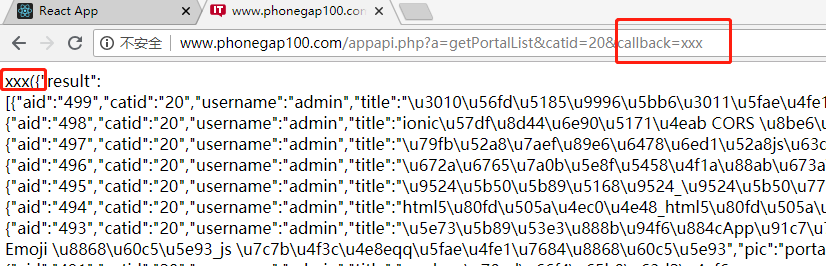
}).catch(function(ex) {

console.log('parsing failed', ex)

})

注意：

当想测试某个链接是否支持跨域请求时，可以在链接后加&callback=xxx并访问，看是否能返回相应的东西。



* 可以发现，axios在请求数据时，会保持链接的原样；fetch-jsonp请求数据时，链接后面会加上&callback=xxx

# 13

React生命周期函数：

组件加载之前，组件加载完成，以及组件更新数据，组件销毁。触发的一系列的方法 ，这就是组件的生命周期函数

* 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

# 14

React静态路由的配置：

1、找到官方文档 <https://reacttraining.com/react-router/web/example/basic>(在github上搜react-router)

2、安装 cnpm install react-router-dom --save

3、找到项目的根组件引入react-router-dom

import { BrowserRouter as Router, Route, Link } from "react-router-dom";

4、复制官网文档根组件里面的内容进行修改 （加载的组件要提前引入）

<Router>

<Link to="/">首页</Link>

<Link to="/news">新闻</Link>

<Link to="/product">商品</Link>

<Route exact path="/" component={Home} />

<Route path="/news" component={News} />

<Route path="/product" component={Product} />

</Router>

exact表示严格匹配， 如果你家exact时，点击“首页”，不仅会匹配Home组件，还会匹配News和Product组件。

# 15

* react动态路由传值

1、动态路由配置

<Route path="/content/:aid" component={Content} />

2、对应的动态路由加载的组件里面获取传值

this.props.match.params

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

* react get传值

<Route path="/Content" component={Content} />

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

1. 获取 this.props.location.search

# 16

react解析html

https://reactjs.org/docs/dom-elements.html

<div className="p\_content" dangerouslySetInnerHTML={{\_\_html: this.state.list.content}}> </div>

# 18

实现js跳转路由：https://reacttraining.com/react-router/web/example/auth-workflow

1、要引入Redirect

import {

BrowserRouter as Router,

Route,

Link,

Redirect,

withRouter

} from "react-router-dom";

2、定义一个flag

this.state = {

loginFlag:false

};

3、render里面判断flag 来决定是否跳转

if(this.state.loginFlag){

return <Redirect to={{ pathname: "/" }} />;

}

4、要执行js跳转

通过js改变loginFlag的状态

改变以后从新render 就可以通过Redirect自己来跳转

# 22

在使用ant design的时候，当设置高级配置，试图让资源按需加载时，启动项目npm start时会报错。react-app-rewired报错

* 原因如下：

react-scripts 2.1.2+现在只有一个webpack.config.js（prod + dev是configs合二为一）。 这会打破react-app-rewired，因为它会查找webpack.config.prod.js

我知道这个模块不再维护，但到目前为止（react-scripts 2.1.1或更早版本）它仍然允许您手动修改配置的部分而不会出现问题。

the new version of react-app-rewired removed all the methods like injectBabelPlugin .  
these methods are moved into a new package called 'customize-cra' , which depends on react-app-rewired@2.x .

* 解决方法：

make sure u installed 'customize-cra' and react-app-rewired@2.x  
make sure u installed 'less' and 'less-loader'

and then here is the config-overrides.js

const {

override,

fixBabelImports,

addLessLoader,

} = require("customize-cra");

module.exports = override(

fixBabelImports("import", {

libraryName: "antd", libraryDirectory: "es", style: true // change importing css to less

}),

addLessLoader({

javascriptEnabled: true,

modifyVars: { "@primary-color": "#1DA57A" }

})

);