官方网址:

<https://react.docschina.org/docs/getting-started.html>

简洁:

React 是一个用于构建用户界面的 JAVASCRIPT 库。React 拥有较高的性能，代码逻辑非常简单，于2013开源。

环境搭建:

1/确保你安装了较新版本的 [Node.js](https://nodejs.org/en/" \t "https://react.docschina.org/tutorial/_blank) >=8.10

node --version

2/确保你安装了较新版本的 npm >=5.6

npm -v

3/使用 “npx create-react-app 项目名” 来建立项目

3.1/由于国外镜像太慢,建议改成国内淘宝镜像

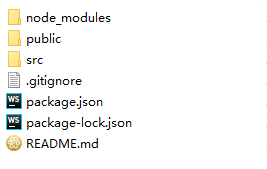
使用npm config get registry ,查看是否是默认 https://registry.npmjs.org/

3.2/修改为淘宝镜像 npm config set registry https://registry.npm.taobao.org

4/进到项目目录下,使用”npm start”运行项目,然后在浏览器访问 http://localhost:3000

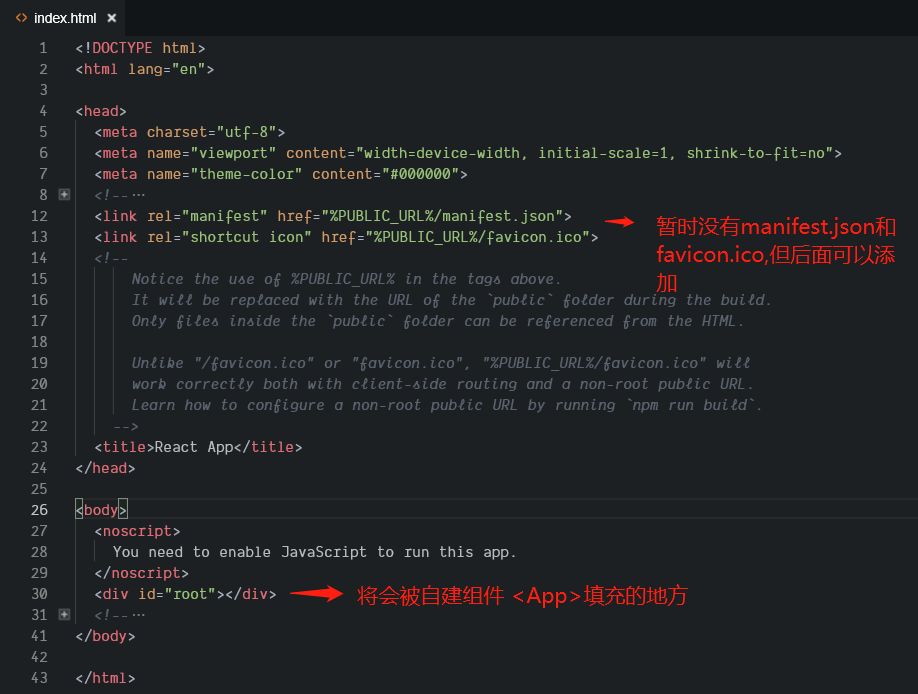
~~5/删掉新项目中src/文件夹下的所有文件~~

目录结构:



node\_modules依然是3方js包存放文件夹.

public/index.html (必须存在,且文件名不可更改)页面模板,index.html是单页面应用的基本模板，所有react生成的代码都会注入到此HTML中.且只有public下的资源能被index.html的%PUBLIC\_URL%(%PUBLIC\_URL%,指代的是public文件夹路径)调用使用,



src/index.js (必须存在,且文件名不可更改)js入口,注意:import App from “./App.js”简写成了import App from “./App”,省略了.js后缀. 或者将App.js放到文件夹App下,写成import App from “./App/App”,或者,在App文件夹下,将App.js改为index.js,写成import App from “./App,此时是可以完全省略了App.js





启动流程:

显示加载public中的index.html,然后加载src中的index.js

极简使用方法:

1/添加”容器”标签.打开你想要编辑的 HTML 页面。添加一个空的 <div> 标签作为标记你想要用 React 显示内容的位置

<!-- ... 其它 HTML ... -->

<div id="like\_button\_container"></div>

<!-- ... 其它 HTML ... -->

给这个 <div> 加上唯一的 id HTML 属性。这将允许我们稍后用 JavaScript 代码找到它，并在其中显示一个 React 组件.React 会替换 DOM 容器内的任何已有内容。

2/添加 Script 标签.将搭建react的脚本引进来.

<!-- ... 其它 HTML ... -->

<!-- 加载 React。-->

<!-- 注意: 部署时，将 "development.js" 替换为 "production.min.js"。-->

<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>

<!-- 加载我们的 React 组件。-->

<script src="like\_button.js"></script>

</body>

3/创建一个react组件

在 HTML 页面文件的同级目录下创建一个名为 like\_button.js 的文件。这个文件可以是我们自定义的组件,也可以是github上的组件.

在 like\_button.js 的底部，加入以下两行代码。

const domContainer = document.querySelector('#like\_button\_container');

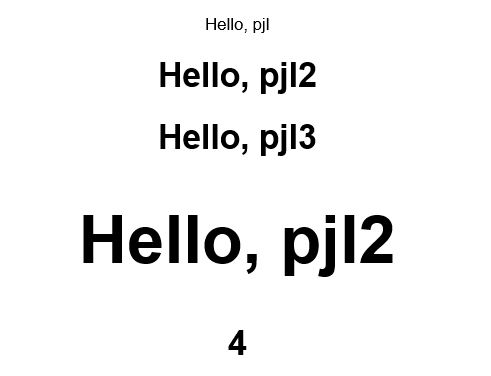
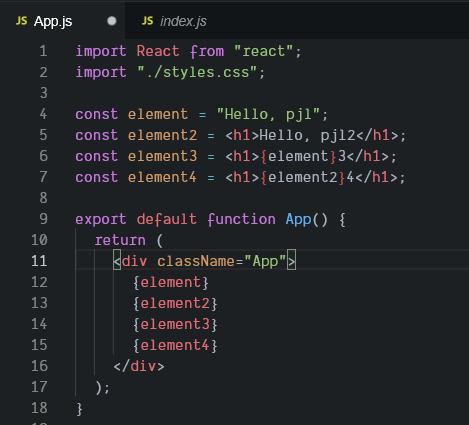
ReactDOM.render(e(LikeButton), domContainer);

这两行代码会找到我们在步骤 1 中添加到 HTML 里的 <div>，然后在它内部显示我们的 React 组件 “Like” 按钮。

jsx是啥:

JSX是JavaScript XML,是React提供的Syntax Sugar(语法糖), 能让我们可以在JS中写html标记语言。这也就解释了,为什么你会在有的使用了react项目中,看到超出你认识的js代码.所有的语法糖都是为了书写方便,或阅读简洁,但这些基于你知道它的”规则”的前提下.

它的典型形式如下:



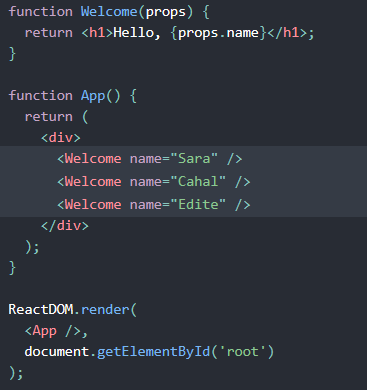
js文件中写着标签.

由于JSX 语法上更接近 JavaScript 而不是 HTML，所以 React DOM 使用 camelCase（小驼峰命名）来定义属性的名称，而不使用 HTML 属性名称的命名约定。如:html中的属性class 变成了 [className](https://developer.mozilla.org/en-US/docs/Web/API/Element/className" \t "https://react.docschina.org/docs/_blank)，而 tabindex 则变为 [tabIndex](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/tabIndex" \t "https://react.docschina.org/docs/_blank),等等.

函数组件和class组件:

哪种方式编写更好?倾向于函数式,因为js本身具备函数式编程的能力.而面向对象的编程,状态问题和类型爆炸一直是bug的根源.

函数组件形式



注意:

1/函数组件,不要用this

2/每个function 函数,都可以作为 <>标签,这和class组件形式的区别还是比较明显

3/用作组件的function函数,必须是Xxx的大写开头

class组件形式



注意红框标出的注意点.

1/涉及到给标签传值,要注意 constructor(props)方法,并且一定要写 super(props).

2/jsx的编写区域限定在了render()渲染方法内,并由return返回,注意return后是(),而不是{}

3/props.name改为了this.props.name,因为class化后,props特指该对象的props属性

4/class化的组件得继承Component

5/所有的render中的return,所有标签应该包括在一个总的标签下,如<div>或<fragment>

6/特别注意,如果以一个class组件一个文件的形式,在该文件最后,一定要加上

export default 这个class名; 来暴露这个组件为全局可用才行,且其他class要使用该组件,得import 组件名 from “组件class文件所在位置(文件后缀可用省略)”

7/只有class声明的才算组件

元素(标签/方法)和组件:

函数组件的一个元素.

const element = <div tabIndex="0">abc</div>

或

const handleClick = () => { setTimeout(showMessage, 3000); };

class组件形式,就是class中的各种方法.

组件是由元素组建起来的,组件非常简单。我们可以认为它们就是简单的函数，接受 props 和 state (后面会讨论) 作为参数，然后渲染出 HTML。注意:**组件名称必须以大写字母开头。组件中可以嵌套嵌套组件**

函数组件形式的一个组件.

// 普通函数式

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

// 箭头函数式

const Title = props => <h1>Hello, {props.name}</h1>

class组件形式的,肯定就是class,并是继承自Component

props属性:

函数组件形式:

class组件形式:

每个继承自React.Component的组件,都自带props属性.它代表使用标签时传给标签的值.

如上图的Welcome的name.

每个组件都有另一个属性:defaultProps来给自定义属性设置默认值.



每个组件都有另一个属性:propTypes来给自定义属性设置数据类型.



父组件向子组件传递,可以通过props,子组件通过this.props.XXX获得

子组件向父组件传递,当然还是通过props来传递,只是此时子组件是给父组件的回调函数传入结果为参数,让父组件使用获得的参数,来改变

State对象和setState()方法:

由于参数“props”只读的特性,对于一些动态灵活的控件,如实时显示时间,我们得用到另一个React为我们准备好的私有对象 “state”,当然,我们得遵守不直接使用 this.state.comment=”hello”的方式来修改它,而是得用内部方法setState({comment:”hello”})来改变.它有什么好处?调用setState会自动触发更新,但出于性能考虑,更新是异步的.

补充:

1/只有set时采用setState,获取时,直接this.state.XXX就好

2/设置state中的数组的某个项:(不要存在只改一项的幻想,只能给新的array)

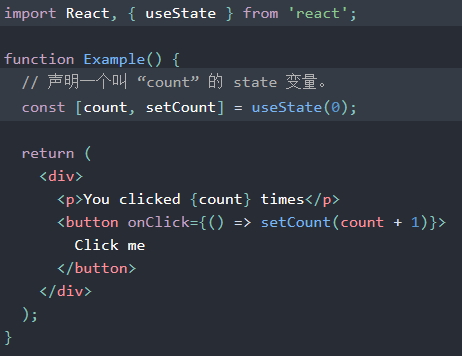
setState({ ‘array’ : this.state.array.concat([neItem])})

函数组件形式:

原本函数式没有state的,但是16.8版本后提供了hook,可以使用

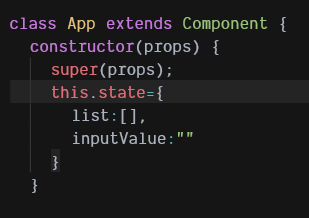
const [要保存在state中的值或函数 , 对state中的值或函数进行加工和进行改变,并替换state值]=useState(初始值 或 函数).如:

显示一个计数器。当你点击按钮，计数器的值就会增加



class组件形式:

state值的初始化,注意是this.state={...}



组件的生命周期:

函数组件形式:

原本函数式,没有明确的生命周期,但16.8后的hook,可以使用

useEffect(函数方法),可以把 useEffect Hook 看做 componentDidMount，componentDidUpdate 和 componentWillUnmount 这三个函数的组合。

class组件形式:

constructor(props)----构造函数

render()----渲染.当ReactDOM.render()调用时,会触发使用的组件的render()方法.有个坑点,一般会发现执行多次

componentWillMount()----渲染前调用

componentDidMount()----组件第一次被渲染(挂载|呈现)到 DOM 中的时候

componentWillUnmount()----组件被删除(卸载)的时候

componentWillReceiveProps()----在组件接收到一个新的 prop (更新后)时被调用。这个方法在初始化render时不会被调用。

shouldComponentUpdate()----在组件接收到新的props或者state时被调用。在初始化时或者使用forceUpdate时不被调用。

componentWillUpdate()----在组件接收到新的props或者state但还没有render时被调用。在初始化时不会被调用。

componentDidUpdate()----在组件完成更新后立即调用。在初始化时不会被调用。

其他内置方法:

设置状态：setState(object nextState[, function callback])

//nextState 是要更新的属性,可以是function或集合.

//callback 是setState方法成功后,组件渲染后的回调

替换状态：replaceState(object nextState[, function callback])

设置属性：setProps(object nextProps[, function callback])

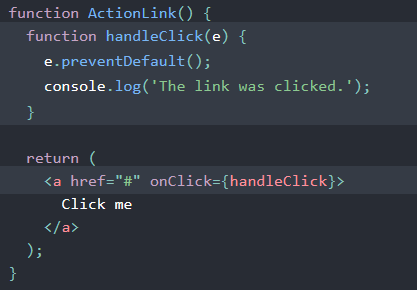
替换属性：replaceProps(object nextProps[, function callback])

强制更新：forceUpdate([function callback])

获取DOM节点：findDOMNode

~~判断组件挂载状态：isMounted(ES6废除)~~

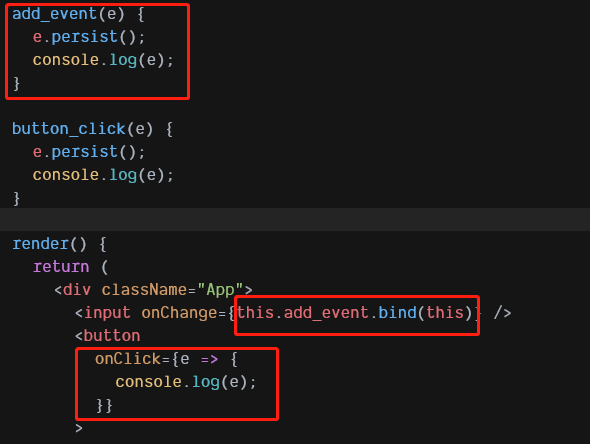
事件:



也就是说,标签的事件属性,会自带一个参数 e .

叫做SyntheticEvent实例

<https://react.docschina.org/docs/events.html>



注意:

1/使用bind绑定事件处理函数时,不需要写成 this.add\_event.bind(this,e),这样反而会报错.因为函数方法的e参数,是react给于的.不需要程序员干预

2/箭头函数表达式,着注意要加上e

3/class组件形式时,注意要加上this,而函数组件形式,不需要加this

事件问题:

当使用事件时出现大量”Warning: This synthetic event is reused for performance reasons. If you're seeing this, you're accessing the property target on a released/nullified synthetic event. This is set to null. If you must keep the original synthetic event around, use event.persist(). See [https://fb.me/react-event-pooling](https://fb.me/react-event-pooling" \t "https://www.jianshu.com/p/_blank) for more information.”时.我们需要在事件的处理方法开头,添加 e.persist()

react中的list操作(也包含map,因为js中的map和list其实是一个东西):

初始化: const list=[“111”,”222”,”333”]

增加:list=[...list,”444”]

删:

改:

查:

遍历--list.map(item =>console.log(item)) 或 list.map((子体,下标)=>console.log(子体item))

注意:每个数组,都需要分配key,即在标签中添加,如遍历的时候一个一个添加

list.map(item => <li key={item}>{item}</li>);

或在总标签一次性添加

<Lists list={list}>

示例:一个增加和删除list元素

import React, { useState } from "react";

export default function App() {

const [value, get\_value] = useState(0);

// const map2 = new Map();//也可以使用Map()对象

//注意,不要写成 const [list, add\_item, del\_item],一个state,只能有一个方法

const [list, setList] = useState([

{ id: 1, list\_value: "111111" },

{ id: 2, list\_value: "222222" },

{ id: 3, list\_value: "333333" }

]); //不要单纯的使用数组["111","222","333"],这样CURD不方便

function button\_event(e) {

e.persist();

console.log(e);

console.log(value);

//list的add

setList([...list, { id: list.lenght + 1, list\_value: value }]);

}

function Getitem() {

//注意被用作b自定义标签的组件,要首字母大写

const item\_click = e => {

e.persist();

console.log(e);

console.log(e.\_targetInst.key); //通过console输出,只有这样才能拿到标签间的内容

const temp = list.splice(e.\_targetInst.key, 1); //list的删除

console.log(temp); //splice返回的是被删除的数据

console.log(list); //list剩下的才是需要的数据

setList([...list]); //直接写成setList(list)是没有效果的,不知道为什么

};

//这里uls表示的是多个<li>标签,使用map来遍历,item子体,index下标

const uls = list.map((item, index) => (

<li onClick={item\_click} key={index}>

{/\*\*注释生效\*\*/ item.list\_value /\*\*拿到list值\*\*/}

</li>

));

return <ul>{uls}</ul>;

}

return ( <div className="App">

<input onChange={e => get\_value(e.target.value)} />

<button onClick={button\_event}>add</button>

<Getitem /> </div>

);

}