视频链接: https://pan.baidu.com/s/1EOGUHzH4eNIIg1mvQRyEPw 密码: b57f

React16.4 开发简书项目之从零基础入门到实战

Ps：如链接不可用，可以联系我 微信：QQQ466125 备注：react视频

我的代码路径：D:\node\koa2\my-app

## 1.0导学



### 1.1设计的知识点



### 1.2学习前提

**Js es6 webpack npm**



### 1.3课程收获



## 2.0介绍react



中文文档 https://react.docschina.org/

英文文档 https://reactjs.org/

### 2.1 react 开发环境准备

引入.js文件来使用react

通过脚手架根据来编码

官方脚手架:Create-react-app //我们用的是这个

### 2.2安装脚手架

官方地址：<https://react.docschina.org/docs/hello-world.html>

英文地址 <https://reactjs.org/docs/add-react-to-a-website.html>

命令行运行:

npm install -g create-react-app //下载脚手架

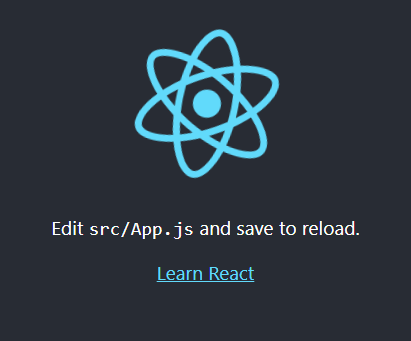
create-react-app my-app //创建项目

cd my-app //进入新创建的my-app这个文件夹下

npm start //运行项目

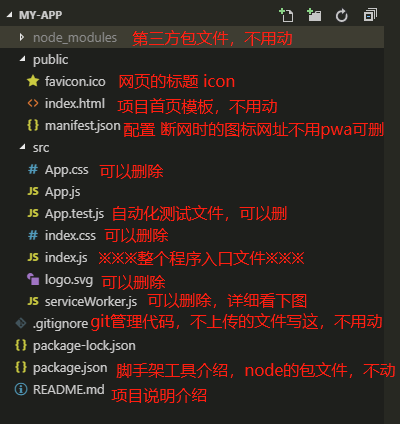
(需要安装node，安装步骤不描述，自行百度就行)

运行成功，会打开http://localhost:3000/，如下图



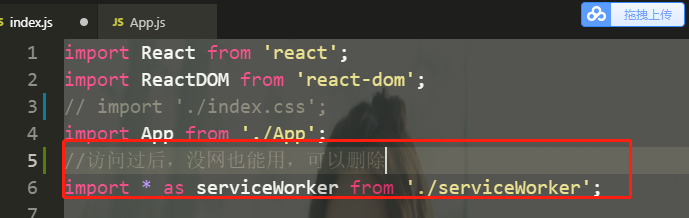
写文档此时我的版本是16.6.2

### 2.3工程目录简介



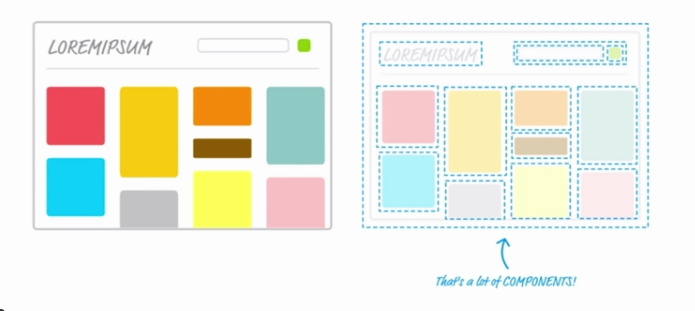
App.test 文件的说明：

访问过后，没网也能用，暂时不用，可以删除



### 2.4 react中的组件

组件的概念



**App.js 这个组件**

import React, { Component } from 'react';

*class* App extends *Component* {

render() {

return (

<div>

hello world

</div>

);

}

}

export default App;

**注：如果想运行jsx 的语法 就需要引用react**

**下面是两个jsx的语法**

ReactDOM.render(<*App* />

render() {

return (

<div>

hello dell lee

</div>

);

}

**Ps:后面代码越来越多，我就放代码就行了。具体的看视频就好 我的react版本是16.63 视频的版本是16.4 差别不大**

### 2.5 react中最基础的jsx语法

Ps：我的网盘这节课视频报错，看不了 跳过就行。

## 3.0 react基础精讲

### 3.1使用react编写TodoList功能

Index.js 文件 修改app 为todoList，

import React from 'react';

import ReactDOM from 'react-dom';

import TodoList from './TodoList';

ReactDOM.render(<*TodoList* />, document.getElementById('root'));

并在src下创建TodoList.js 文件

import React , { Component } from 'react';

*class* TodoList extends *Component* {

render() {

return (

<div><input /> <button>提交</button></div>

<ul>

<li>学英语</li>

<li>learning React</li>

</ul>

)

}

}

export default TodoList;

**这个时候会报错，意料之中**



**Render 函数返回的 只能有一个元素，so，我们在外面包一层div**

<div>

<div><input /> <button>提交</button></div>

<ul>

<li>学英语</li>

<li>learning React</li>

</ul>

</div>

**页面如下：**



**如果想要页面上 第一个元素就是div 和ul 而不展示最外层包的哪个div，可以引用fragment，用fragment 替代div就行**

import React , { Component, Fragment } from 'react';

*class* TodoList extends *Component* {

render() {

return (

<*Fragment*>

<div><input /> <button>提交</button></div>

<ul>

<li>学英语</li>

<li>learning React</li>

</ul>

</*Fragment*>

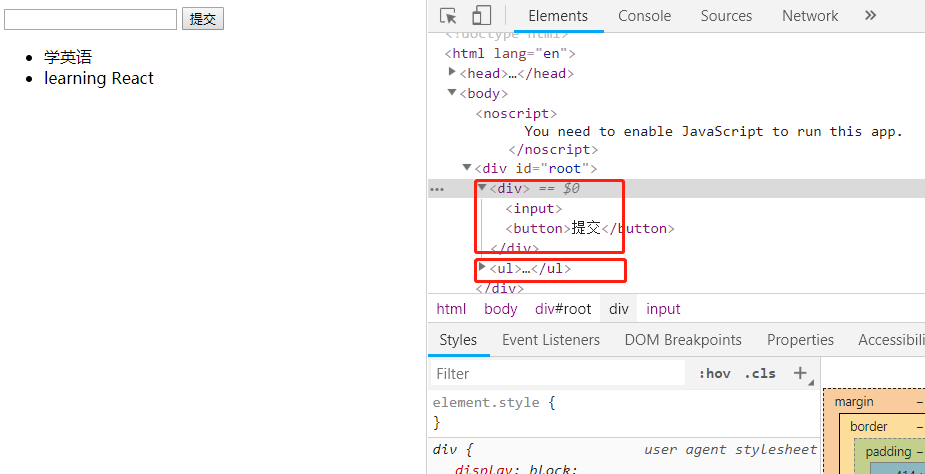
)

}

}

export default TodoList;

**效果图，没报错，并且root下直接就是div+ul**



Ps：现在我们要实现一个功能，输入框的内容，点击提交 就显示到下面的别里面，请看下节详细介绍

### 3.2 react中响应式设计思想和事件绑定（重要）

import React , { Component, Fragment } from 'react';

*class* TodoList extends *Component* {

*constructor*(*props*) {

super(props);

this.state = {

inputValue: 'hello',

list: []

}

}

render() {

return (

<*Fragment*>

<div>

<input

value={this.state.inputValue}

onChange={this.handleInputChange.bind(this)}

/>

<button>提交</button>

</div>

<ul>

<li>学英语</li>

<li>learning React</li>

</ul>

</*Fragment*>

)

}

handleInputChange(*e*){

this.setState({

inputValue: e.target.value

})

}

}

export default TodoList;

### 3.3 实现TodoList新增删除功能

iimport React , { Component, Fragment } from 'react';

*class* TodoList extends *Component* {

*constructor*(*props*) {

super(props);

this.state = {

inputValue: '',

list: []

}

}

render() {

return (

<*Fragment*>

<div>

<input

value={this.state.inputValue}

onChange={this.handleInputChange.bind(this)}

/>

<button onClick={this.handleBtnClick.bind(this)} >提交</button>

</div>

<ul>

{

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

return <li key={index} onClick={this.handleitemDelete.bind(this,index)}>{item}</li>

})

}

</ul>

</*Fragment*>

)

}

handleInputChange(*e*){

this.setState({

inputValue: e.target.value

})

}

handleBtnClick(){

this.setState({

list:[...this.state.list, this.state.inputValue],

inputValue:''

})

}

handleitemDelete(*index*){

//imutable

//state 不允许我们做任何改变

*const* list = [...this.state.list]

list.splice(index,1)

*console*.log(index)

this.setState({

list:list

})

}

}

export default TodoList;

### 3.4 jsx语法细节补充

首先我们在src文件下新建一个style.css的css文件

代码如下

.input{

*border*: 1px solid red;

}

然后再 TodoList 里面引入css

import React , { Component, Fragment } from 'react';

import './style.css'

*class* TodoList extends *Component* {

*constructor*(*props*) {

super(props);

this.state = {

inputValue: '',

list: []

}

}

render() {

return (

<*Fragment*>

<div>

<label htmlFor='inserArea'>输入内容</label>

<input

className='input'

id="inserArea"

value={this.state.inputValue}

onChange={this.handleInputChange.bind(this)}

/>

<button onClick={this.handleBtnClick.bind(this)} >提交</button>

</div>

<ul>

{

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

return <li key={index} onClick={this.handleitemDelete.bind(this,index)}

dangerouslySetInnerHTML={{\_\_html:item}}

>

</li>

})

}

</ul>

</*Fragment*>

)

}

handleInputChange(*e*){

this.setState({

inputValue: e.target.value

})

}

handleBtnClick(){

this.setState({

list:[...this.state.list, this.state.inputValue],

inputValue:''

})

}

handleitemDelete(*index*){

//imutable

//state 不允许我们做任何改变

*const* list = [...this.state.list]

list.splice(index,1)

*console*.log(index)

this.setState({

list:list

})

}

}

export default TodoList;

Ps ： 这里注意 label里 for 的用法 应该是htmlFor   
 class 应该是 className

### 3.5 拆分组件与组件之间的传值

我们再src 下新建一个TodoItem.js的文件，代码如下

import React,{ Component } from 'react';

*class* TodoItem extends *Component* {

*constructor*(*props*) {

super(props);

this.handleClick = this.handleClick.bind(this)

}

render() {

return (

<div onClick={this.handleClick}>

{this.props.content}

</div>)

}

handleClick() {

this.props.deleteItem(this.props.index)

// alert(this.props.index);

}

}

export default TodoItem;

父组件向子组件 通过属性传值 比如content

子组件接受 this.props.content

Ps：这节课我出错了两次，都是因为代码写错了单词。。。

报错的时候会提醒，哪错改哪就行

TodoList.Js代码如下

import React , { Component, Fragment } from 'react';

import './style.css'

import TodoItem from './TodoItem'//引入todoitem

*class* TodoList extends *Component* {

*constructor*(*props*) {

super(props);

this.state = {

inputValue: '',

list: []

}

}

render() {

return (

<*Fragment*>

<div>

<label htmlFor='inserArea'>输入内容</label>

<input

className='input'

id="inserArea"

value={this.state.inputValue}

onChange={this.handleInputChange.bind(this)}

/>

<button onClick={this.handleBtnClick.bind(this)} >提交</button>

</div>

<ul>

{

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

return (

<div>

<*TodoItem*

content={item}

index={index}

deleteItem={this.handleitemDelete.bind(this)}

/>

</div>

)

})

}

</ul>

</*Fragment*>

)

}

handleInputChange(*e*){

this.setState({

inputValue: e.target.value

})

}

handleBtnClick(){

this.setState({

list:[...this.state.list, this.state.inputValue],

inputValue:''

})

}

handleitemDelete(*index*){

*const* list = [...this.state.list]

list.splice(index,1)

this.setState({

list:list

})

}

}

export default TodoList;

### 3.6 TodoList代码优化

为了提升代码的性能，我们对代码进行精简，并且把js的key报错给解决掉

TodoItem.js代码

import React,{ Component } from 'react';

*class* TodoItem extends *Component* {

*constructor*(*props*) {

super(props);

this.handleClick = this.handleClick.bind(this)

}

render() {

*const* { content } = this.props

return (

<div onClick={this.handleClick}>

{content}

</div>

)

}

handleClick() {

*const* {deleteItem,index} =this.props

deleteItem(index)

}

}

export default TodoItem;

TodoList.js代码如下

import React , { Component, Fragment } from 'react';

import TodoItem from './TodoItem'//引入todoitem

import './style.css'

*class* TodoList extends *Component* {

*constructor*(*props*) {

super(props);

this.state = {

inputValue: '',

list: []

}

this.handleInputChange = this.handleInputChange.bind(this);

this.handleBtnClick = this.handleBtnClick.bind(this);

this.handleItemDelete = this.handleItemDelete.bind(this);

}

render() {

return (

<*Fragment*>

<div>

<label htmlFor='inserArea'>输入内容</label>

<input

className='input'

id="inserArea"

value={this.state.inputValue}

onChange={this.handleInputChange}

/>

<button onClick={this.handleBtnClick} >提交</button>

</div>

<ul>

{this.getTodoItem()}

</ul>

</*Fragment*>

)

}

getTodoItem() {

return this.state.list.map((*item*, *index*) *=>* {

return (

<*TodoItem*

key={index}

content={item}

index={index}

deleteItem={this.handleItemDelete}

/>

)

})

}

handleInputChange(*e*) {

*const* value = e.target.value

this.setState(() *=>* ({

inputValue: value

}))

}

handleBtnClick(){

this.setState((*prevState*) *=>*({

list:[...prevState.list, prevState.inputValue],

inputValue:''

}))

}

handleItemDelete(*index*){

this.setState((*prevState*) *=>* {

*const* list = [...prevState.list];

list.splice(index,1);

return { list }

});

}

}

export default TodoList;

### 3.7 围绕react衍生出的思考

**1.声明式开发**  对比 Jq js命令式开发

**2.可以与其它框架并存** 例如vue

**3.组件化**

**4.单向数据流**

**5.视图层框架**

**6.函数式编程 //更容易自动化测试**

## 4.0react高级内容

4.1reactdevlopertools安装及使用