react

2018年11月14日 星期三　　上午 9:08

1.项目代码中别名的设置

在.config.js文件中的resolve中的alias中设置

2.react调试工具

React Developer Tools调试工具

[React Developer Tools - Chrome 扩展下载安装地址](https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=zh-CN)

3.ES6展开运算符 ...

在JSX中使用js语法，要使用{}包裹

{...user}表示user中的所有属性

4.class继承，类

1.

2.super()

```
// 这是父类 【大家可以直接把 父类，理解成 原型对象 prototype】
class Person {
  constructor(name, age){
    this.name = name
    this.age = age
  }

  // 打招呼 的 实例方法
  sayHello(){
    console.log('大家好')
  }
}
// 这是子类 美国人
// 在 class 类中，可以使用 extends 关键字，实现 子类继承父类
// 语法：  class 子类 extends 父类 {}
class American extends Person {
  constructor(name, age){
    // 问题1：为什么一定要在 constructor 中调用 super
    //   答案：  因为，如果一个子类，通过 extends 关键字继承了父类，那么，在子类的 constructor 构造函数中，必须 优先调用一下
    // 问题2：super 是个什么东西？
    //   答案：  super 是一个函数，而且，它是 父类的 构造器；子类中的 super,其实就是父类中，constructor 构造器的一个引用；
    // 问题3：为什么 调用了 super() 之后，a1 实例的 name 和 age 都变成 undefined 了？
    super(name, age)
  }
}
const a1 = new American('Jack', 20)
console.log(a1)
a1.sayHello()
```

3.class创建组件

```
import React from 'react'
import ReactDOM from 'react-dom'

// 使用 ES6中 import 导入需要的组件
// import Hello from '@/components/Hello'

// 导入 class 继承
// import '@/03.class-继承-公共方法'
// class 关键字创建组件
class Movie extends React.Component {
  // 构造器
  constructor() {
    // 由于 Movie 组件，继承了 React.Component 这个父类，所以，自定义的构造器中，必须 调用 super()
    super()
    // 只有调用了 super() 以后，才能使用 this 关键字
    this.state = { // 这个 this.state = {} 就相当于 Vue 中的 data() { return { } }
      msg: '大家好，我是 class 创建的 Movie组件'
    }
  }
  // render 函数的作用，是 渲染 当前组件所对应的 虚拟DOM元素
  render() {
    // return null
    // 在 class 关键字创建的组件中，如果想使用 外界传递过来的 props 参数，不需接收，直接通过 this.props.*** 访问即可
    // 注意：不论是 class 还是普通 function 创建的组件，它们的 props 都是只读的！
    // this.props.name = '李四'
    // 在 class 创建的组件中， this.state 上的数据，都是可读可写的！
    // this.state.msg = 'msg的值被我修改了！'
    return <div>
      {/* 注意：在 class 组件内部，this 表示 当前组件的实例对象 */}
      这是 Movie 组件 -- {this.props.name} -- {this.props.age} -- {this.props.gender}
      <h3>{this.state.msg}</h3>
    </div>
  }
}

// 使用ES6 export default 向外暴露成员
export default Movie
```

4.function创建组件和class创建组件的区别

```
// 在 webpack 中，每一个 单独的 JS 文件或 JSX 文件，都有自己独立的作用域：默认，外界无法访问 模块中的成员；
export default function Hello(props) {
  console.log(props)
  return <div>
    这是 用 function 构造函数创建的组件 --{props.name} --
    {props.age} -- {props.gender}
  </div>
}

// 使用ES6 export default 向外暴露成员
// export default Hello
```

```
// class 关键字创建组件
class Movie extends React.Component {
  // render 函数的作用，是 渲染 当前组件所对应的 虚拟DOM元素
  render() {
    // return null
    // 在 class 关键字创建的组件中，如果想使用 外界传递过来的 props 参数，不需接收，直接通过 this.props.*** 访问即可

    // 注意：不论是 class 还是普通 function 创建的组件，它们的 props 都是只读的！
    // this.props.name = '李四'

    return <div>
      {/* 注意：在 class 组件内部，this 表示 当前组件的实例对象 */}
      这是 Movie 组件 -- {this.props.name} -- {this.props.age}
      -- {this.props.gender}
    </div>
  }
}
```

5.子向父组件传值

用的是方法的调用

1.在父组件上定义一个函数

```
// 演示 子向父 传值
parentShow = (val) => {
    console.log('有人调用了 父组件上的方法；总有刁民想害朕！' + val)
    this.setState({
    sonVal: val
    })
}
```

2.在父组件中，的子组件中 加入一个方法，调用这个函数

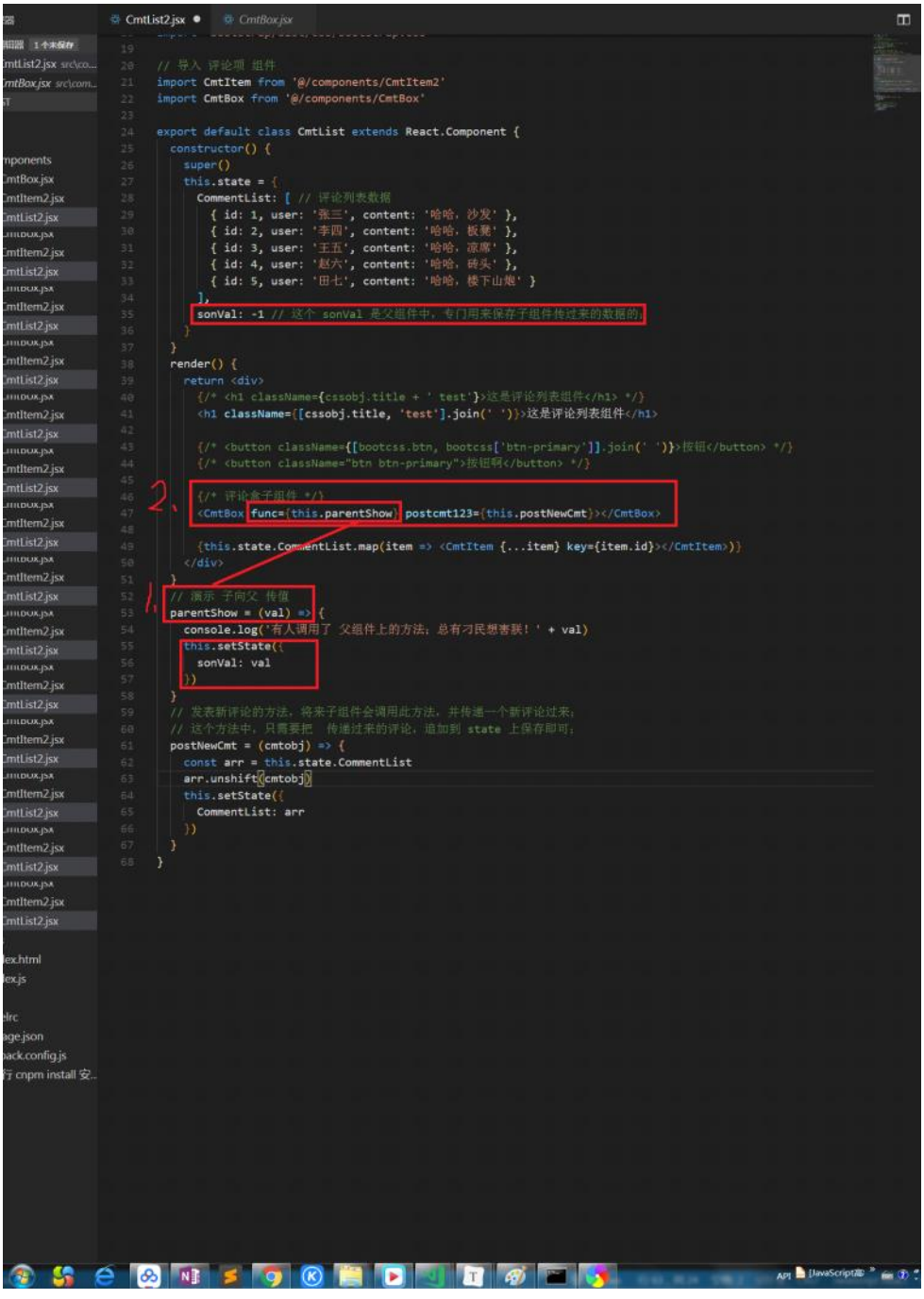
```
{/* 评论盒子组件 */}
<CmtBox func={this.parentShow} postcmt123={this.postNewCmt}></CmtBox>
```

2.在子组件上通过this.props调用func,就是调用了父组件上的方法，就是在父组件中的子组件调用的方法

```
this.props.func(100)，，子向父传值100，也可以换成其他
const cmtobj = { user, content, id: Date.now() }
// 4. 将最新的评论消息，追加 并 保存到 数组中
// 4.1 这是，就涉及到了 子组件 向 父组件传值
// 4.2 注意： 父 向 子 传值，用的是 属性绑定的形式 vue :initcount="10" initcount={10}
// 4.3 注意： 子 向 父 传值，用的是 方法的调用；子组件在调用父组件传递过来的方法的时候，同时传递一个实参过去
// this.props.func(100)
this.props.postcmt123(cmtobj)
```

3.在父组件上保存子组件传递过来的值

```
sonVal: -1 // 这个 sonVal 是父组件中，专门用来保存子组件传过来的数据的；
val是形参，实际是子组件传递过来的值，再对子组件传递的值保存
// 演示 子向父 传值
parentShow = (val) => {
    console.log('有人调用了 父组件上的方法；总有刁民想害朕！' + val)
    this.setState({
        sonVal: val
    })
}
```



```
// 点击按钮 发表评论
postCmt = () => {
  // 分析：
  // 1. 获取到 文本框中的 值
  const user = this.refs.user.value
  const content = this.refs.content.value
  // 2. 校验值 是否为空 如果为空，则 alert 弹框提示并退出后续流程
  if (user.trim().length <= 0 || content.trim().length <= 0) return alert('信息不完整！')
  // 3. 如果验证通过，组出一个评论信息对象
  const cmtobj = { user, content, id: Date.now() }
  // 4. 将最新的评论消息，追加 并 保存到 数组中
  //  4.1 这是，就涉及到了 子组件 向 父组件传值
  //  4.2 注意： 父 向 子 传值，用的是 属性绑定的形式 vue :initcount="10"    initcount={10}
  //  4.3 注意： 子 向 父 传值，用的是 方法的调用，子组件在调用父组件传递过来的方法的时候，同时传递一个实参过去
  // this.props.func(100)
  this.props.postcmt123(cmtobj)
  this.refs.user.value = this.refs.content.value = ''
 }
}
```

6.父向子组件传值

1.父组件

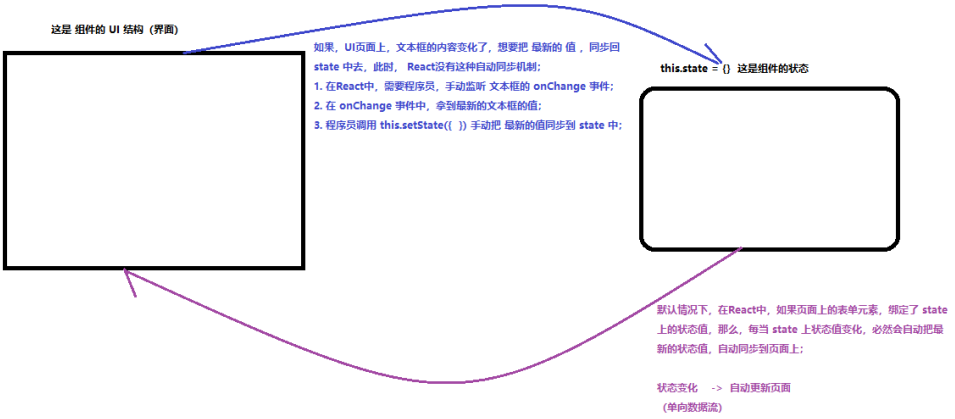
```
this.state = {
  CommentList: [ // 评论列表数据
    { id: 1, user: '张三', content: '哈哈，沙发' },
    { id: 2, user: '李四', content: '哈哈，板凳' },
    { id: 3, user: '王五', content: '哈哈，凉席' },
    { id: 4, user: '赵六', content: '哈哈，砖头' },
    { id: 5, user: '田七', content: '哈哈，楼下山炮' }
  ],
  sonVal: -1 // 这个 sonVal 是父组件中，专门用来保存子组件传过来的数据的；
 }
}
render() {
 return <div>
  {/* <h1 className={cssobj.title + ' test'}>这是评论列表组件</h1> */}
  <h1 className={[cssobj.title, 'test'].join(' ')}>这是评论列表组件</h1>

  {/* <button className={[bootcss.btn, bootcss['btn-primary']].join(' ')}>按钮</button>
  {/* <button className="btn btn-primary">按钮啊</button> */}

  {/* 评论盒子组件 */}
  <CmtBox func={this.parentShow} postcmt123={this.postNewCmt}></CmtBox>

  {this.state.CommentList.map(item => <CmtItem {...item} key={item.id}></CmtItem>)}
 </div>
}
```

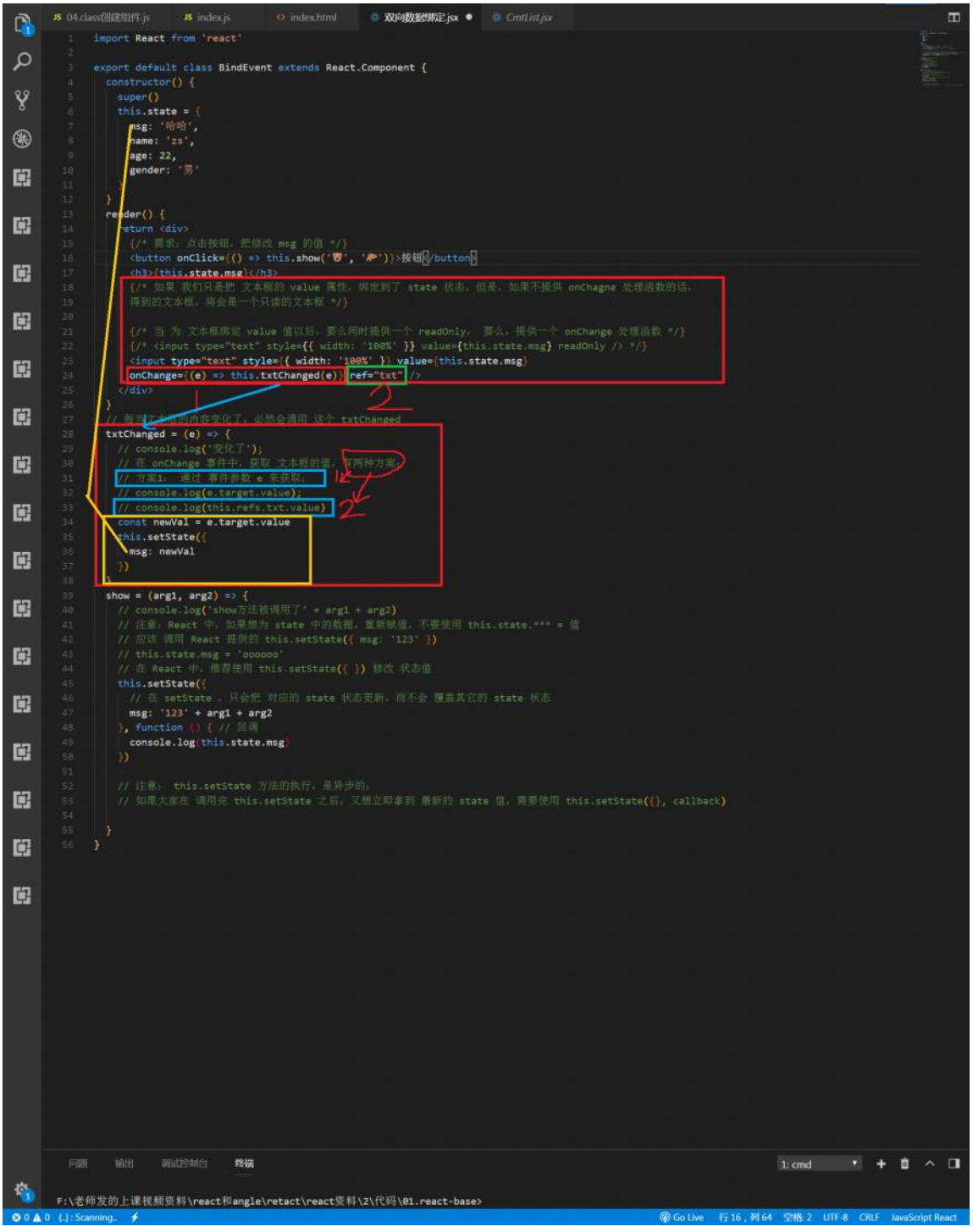
2.子组件通过props接收就可以了

```
import React from 'react'

import cssobj from '@/css/cmtitem.scss'
console.log(cssobj);

export default function CmtItem(props) {
  return <div className={cssobj.cmtbox}>
    <h1 className={cssobj.title}>评论人：{props.user}</h1>
    <p className={cssobj.content}>评论内容：{props.content}</p>
  </div>
}
```

7.双向绑定

```
{/* 如果 我们只是把 文本框的 value 属性，绑定到了 state 状态，但是，如果不提供 onChagne 处理函数的话，得到的文本框，将会是一个只读的文本框 */}
{/* 当 为 文本框绑定 value 值以后，要么同时提供一个 readOnly， 要么，提供一个 onChange 处理函数 */}
{/* <input type="text" style={{ width: '100%' }} value={this.state.msg} readOnly /> */}
<input type="text" style={{ width: '100%' }} value={this.state.msg} onChange={(e) => this.txtChanged(e)} ref="txt" />
```


这是 组件的 UI 结构（界面）
如果，UI页面上，文本框的内容变化了，想要把 最新的 值 ，同步回 state 中去，此时， React没有这种自动同步机制；
1. 在React中，需要程序员，手动监听 文本框的 onChange 事件；
2. 在 onChange 事件中，拿到最新的文本框的值；
3. 程序员调用 this.setState({ }) 手动把 最新的值同步到 state 中；
this.state = {} 这是组件的状态
默认情况下，在React中，如果页面上的表单元素，绑定了 state 上的状态值，那么，每当 state 上状态值变化，必然会自动把最新的状态值，自动同步到页面上；
状态变化 -> 自动更新页面
（单向数据流）


04.class创建组件.js
index.js
index.html
双向数据绑定.jsx
CmtList.jsx
import React from 'react'
export default class BindEvent extends React.Component {
constructor() {
super()
this.state = {
msg: '哈哈',
name: 'zs',
age: 22,
gender: '男'
}
render() {
return
{/* 需求：点击按钮，把修改 msg 的值 */}
<button onClick={() => this.show('🐯', '🐎')}>按钮</button>
<h3>{this.state.msg}</h3>
{/* 如果 我们只是把 文本框的 value 属性，绑定到了 state 状态，但是，如果不提供 onChagne 处理函数的话，
得到的文本框，将会是一个只读的文本框 */}
{/* 当 为 文本框绑定 value 值以后，要么同时提供一个 readOnly， 要么，提供一个 onChange 处理函数 */}
{/* <input type="text" style={{ width: '100%' }} value={this.state.msg} readOnly /> */}
<input type="text" style={{ width: '100%' }} value={this.state.msg}
onChange={(e) => this.txtChanged(e)} ref="txt" />

}
// 每当文本框的内容变化了，必然会调用 这个 txtChanged
txtChanged = (e) => {
// console.log('变化了');
// 在 onChange 事件中，获取 文本框的值，有两种方案：
// 方案1： 通过 事件参数 e 来获取：
// console.log(e.target.value);
// console.log(this.refs.txt.value)
const newVal = e.target.value
this.setState({
msg: newVal
})
}
show = (arg1, arg2) => {
// console.log('show方法被调用了' + arg1 + arg2)
// 注意：React 中，如果想为 state 中的数据，重新赋值，不要使用 this.state.*** = 值
// 应该 调用 React 提供的 this.setState({ msg: '123' })
// this.state.msg = 'oooooo'
// 在 React 中，推荐使用 this.setState({ }) 修改 状态值
this.setState({
// 在 setState ，只会把 对应的 state 状态更新，而不会 覆盖其它的 state 状态
msg: '123' + arg1 + arg2
}, function () { // 回调
console.log(this.state.msg)
})
// 注意： this.setState 方法的执行，是异步的，
// 如果大家在 调用完 this.setState 之后，又想立即拿到 最新的 state 值，需要使用 this.setState({}, callback)
}
}
问题
输出
调试控制台
终端
1: cmd
F:\老师发的上课视频资料\react和angle\retact\react资料\2\代码\01.react-base>
Go Live
行 16，列 64
空格: 2
UTF-8
CRLF
JavaScript React