**WEB第五阶段React —— DAY03**

复习：

1、React是一个用于构建用户界面的JS库 —— 只涉及JS

①声明式 ②组件式开发 ③一次开发到处运行

2、React中自定义组件的两种方式

function式组件：

function MfRating(props){

return (JSX)

}

let e = (<MfRating color="red" count={8}/>)

class式组件：

class MfRating extends React.Component{

render(){

//this.props

return (JSX)

}

}

let e = (<MfRating color="red" count={8}/>)

3、React中的数据绑定

let e = (<div title={表达式} style={表达式} className={表达式} onClick={表达式}>{表达式}</div>)

经典问题：React的事件处理函数中this指向undefined

|  |
| --- |
| 解决方案1：使用箭头函数 —— 调用时不能传参  class App extends React.Component{  f1 = ()=>{  clg(this)  }  render(){  <button onClick={this.f1}></button>  }  } |
| 解决方案2：使用箭头函数 —— 调用时可以传参  class App extends React.Component{  f1(){  clg(this)  }  render(){  <button onClick={()=>this.f1()}></button>  }  } |
| 解决方案3：使用bind  class App extends React.Component{  f1(){  clg(this)  }  render(){  <button onClick={this.f1.bind(this)}></button>  }  } |
| 解决方案4：使用bind  class App extends React.Component{  constructor(){ //构造方法——生命周期方法。调用且仅调用一次，类似于Vue.js中的created( )  super( ) //在构建子类对象之前先构建一个父类对象  this.f1 = this.f1.bind(this)  }  f1(){  clg(this)  }  render(){  <button onClick={ this.f1 }></button>  }  } |

**一、React中的状态变量**

|  |  |  |
| --- | --- | --- |
|  | **声明模型变量** | **状态管理模块** |
| Vue.js | export default {  data(){ //模型数据  return { age: 18 }  }  } | Vuex： state |
| React | class App extends Component{  state = { age: 18 } //模型状态  } | Redux： state |

|  |
| --- |
| 面试题：Vue.js和React有哪些区别？  尤雨溪答：都是MVVM框架，底层都有虚拟DOM概念。但是“数据响应式”方面原理不同：  **Vue.js:**  this.count--，直接修改模型数据，框架会自动重新渲染视图——Push Based：“基于推送的数据响应式”  **React：** this.state.count--，仅仅修改模型数据，但不会通知渲染系统；  this.setState({count: this.state.count-1})，不单修改模型数据，还会通知渲染系统——Pull Based：“基于拉取的数据响应式” |

**React中声明状态变量：**

state = {

age: 18,

uname: 'dangdang',

list: [{}, {}, ...]

}

**使用状态变量：**

<div>{this.state.age}</div>

**修改状态变量：**

this.setState({ age: 19 }) //只修改指定的属性，其它属性会合并

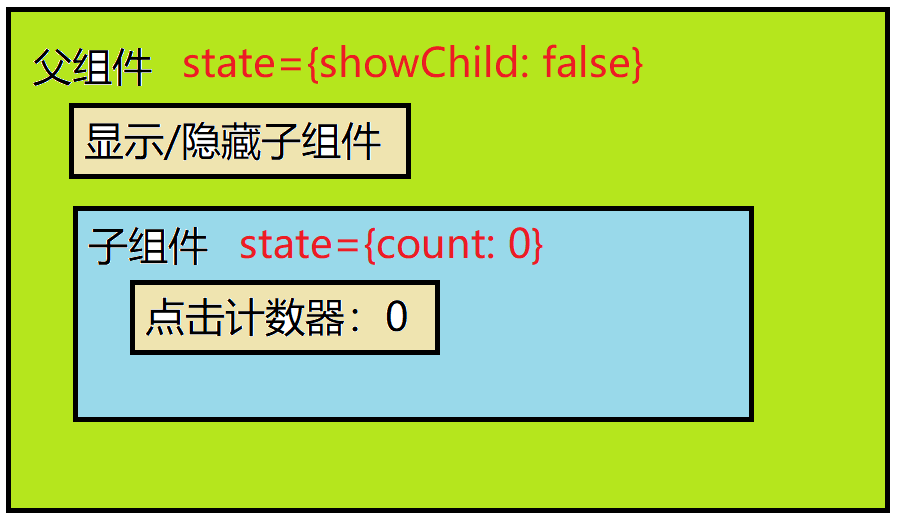
clg(this.state.age) //18

this.setState({ age: 19 }, ()=>{ //setState是异步的，回调函数中才能拿到修改后的数据

clg(this.state.age) //19

})

练习：完成如下的用户界面效果：



|  |
| --- |
| Vue.js中组件的生命周期：  创建阶段： beforeCreate、created  挂载阶段： beforeMount、mounted  更新阶段： beforeUpdate、updated  销毁阶段： beforeDestroy、destroyed |

**二、React中的生命周期方法**

**挂载阶段：**

constructor( ) => render( ) => componentDidMount( )

组件被创建了 => 渲染内容 => 组件完成挂载

**更新阶段：**

shouldComponentUpdate( ) => render( ) => componentDidUpdate( )

应该让组件执行此次更新吗？ => 渲染内容 => 组件完成更新

**卸载阶段：**

componentWillUnmount( )

组件即将卸载

**三、React Hook**

React刚推出时，提供了两种组件语法：

**①function式组件**  function Rating(){ }

**②class式组件**  class Rating extends React.Component{ }

class式组件因为继承了Component，所以获得了很多成员：state、setState、render、componentDidMount...

而function式组件没有继承，也就没有任何属性(state)和方法(生命周期)可用，功能有限，适用的场景很少...

从React V16.8开始，官方为函数式组件添加了很多非常强大的功能——弥补了之前所有的不足！！而且function中还没有class中很恼人的this问题，社区中开始广泛使用function式组件。

V16.8为函数式组件添加的新功能：Hook —— 钩子

**作用：**为函数式组件“钩住”更多的功能，以弥补之前的不足 —— 不能用于类式组件

**本质：**钩子的本质就是一个函数，返回一些特别的对象或数据供组件使用 —— 官方要求，所有的钩子函数都必须以 use 开头，官方提供了15个钩子，不同的钩子函数返回不同的功能

**注意：**钩子函数只能用于函数式组件的最外层！！

**第一个钩子：状态钩子（useState）—— 用于为组件添加“状态变量”以及“修改状态变量的方法”**

let [ 变量名, 修改方法名 ] = useState( 初始值 )

**第二个钩子：副作用钩子（useEffect）—— 用于为组件添加各种不同的“生命周期方法”**

useEffect( function(){ //这就是一个生命周期方法 } )

**四、毕业项目：净美仕企业门户**

课后任务：

①读手册，掌握不同时期的生命周期方法使用 useEffect( ) 该如何实现

②读mfresh项目的早期源代码（纯HTML/CSS + jQuery编写的），了解项目的结构；

如果自己设计开发，那么需要多少个表？多少个接口？多少个页面组件？多少个自定义组件？

www.codeboy.com:9999/mfresh/