**WEB第五阶段React — DAY04**

复习：

1、React中两种自定义组件语法：

function式组件： function MfRating(props){return (JSX)}

class式组件： class MfRating extends React.Component{ render(){return (JSX)}}

2、class组件的状态变量

state = {age: 0, list:[ ]}

//this.state.age = 10 错误

this.setState({age: 10})

this.setState({age: 10}, ()=>{clg(this.state.age)})

3、class组件的生命周期方法

挂载阶段： constructor => render => componentDidMount

更新阶段： shouldComponentUpdate? => render => componentDidUpdate

销毁阶段： componentWillUnmount

4、React Hook

Hook：钩子，本质就是一个use-开头的函数，只能在function组件内（最顶层），用于为函数式组件“钩住”更多的功能。

官方提供的钩子1：useState（状态钩子）

let [age, setAge] = useState(0)

setAge(10)

let [list, setList] = useState([ ])

setList([{},{},...])

官方提供的钩子2：useEffect（副作用钩子）

useEffect( 函数(生命周期函数) )

**一、React Hook：useEffect与函数式组件的生命周期**

|  |  |
| --- | --- |
| **function式组件** | **class式组件** |
| useEffect( ( )=>{  //生命周期方法1=组件挂载+任意数据改变  } ) | componentDidMount  +  componentDidUpdate |
| useEffect( ( )=>{  //生命周期方法2=组件挂载+依赖的数据改变  } , [count, age] ) | componentDidMount  +  componentDidUpdate |
| useEffect( ( )=>{  //生命周期方法3=组件挂载  } , [ ] ) | componentDidMount |
| useEffect( ( )=>{  return ( )=>{  //生命周期方法4=任意数据改变+组件卸载  }  } ) | componentDidUpdate  +  componentWillUnmount |
| useEffect( ( )=>{  return ( )=>{  //生命周期方法5=依赖的数据改变+组件卸载  }  } , [count, age]) | componentDidUpdate  +  componentWillUnmount |
| useEffect( ( )=>{  return ( )=>{  //生命周期方法6=组件卸载  }  } , [ ]) | componentWillUnmount |

|  |
| --- |
| WEB项目架构设计的历史演变：  1991年——第一阶段：纯静态页面，无法访问服务器/数据库  1994年——第二阶段：动态网页开发，先编写.html，重命名为.asp，穿插写入Java代码，运行于服务器——前后端没有分离  1998年——第三阶段：异步请求，前端程序员只写.html，后端程序员只接.java，使用AJAX异步请求——前端代码和后端代码仍然打为一个压缩包，运行在一个服务器——前后端半分离  2010年——第四阶段：前后端完全分离，前端代码运行于一个服务器，后端代码运行于另一个服务器，二者通过“跨域请求”连接  2013年——第五阶段：前后端分离/前端工程化，前端开始有服务器端渲染、脚手架、MVVM框架… |

**二、React阶段项目：净美仕企业官网**

背景：净美仕公司企业门户

2013年，第一版开发完成，采用的是前后端未分离的PHP开发方案，所有的前端代码和后端代码全都打包在一起，发布在一台服务器中。

目前来看，这套技术体系已经过时了。甲方希望采用主流的“前后端分离”技术重构该项目。架构设计为：

**后端数据API：**可以暂时仍然复用之前的PHP代码，后续可以采用Node.js或Golang重构。

**前端架构：**采用React + React-Router-DOM项目脚手架重构。

开发步骤：

0.在本机的XAMPP服务器套件中，运行之前的V1版本的项目

如果本机没有搭建成功，可以访问：www.codeboy.com:9999/mfresh/

1.创建一个React项目，下载必需的第三方模块

npx create-react-app mfresh\_v2

cd mfresh\_v2

npm i react-router-dom //路由模块

npm i redux //React的状态保持模块，类似于Vue.js中的Vuex

npm i react-redux //React和Redux整合需要的模块

2.阅读V1版本的开发文档或代码，创建V2必需的pages和components

3.在index.js中创建路由导航系统

4.详细编写每个页面和内部的组件

Home < MfHeader + MfFooter + MfCarousel

News

NewsDetails

Register

Login

Product

ProductDetails

|  |
| --- |
| 重要知识点： Vue.js / React 项目中图片放置路径？  **方法1：public目录下，**不会被Webpack处理，而是直接发布到最终的Web服务器根目录下，客户端可以通过地址直接访问到该图片！ —— 项目中的“动态图片”（例如头像、评论照片、商品图）只能放在public目录下  **方法2：src目录下**，默认不会被Webpack处理——除非被其它模块文件导入，就会被打包编译到最终发布的成品中；否则最终的成品中就不会出现该图片！—— 项目中的“静态图片”（例如logo、精灵图）可以放到src目录下，在Webpack打包时就可以导入了 |

|  |  |  |  |
| --- | --- | --- | --- |
| 目前学过的路由&导航技术 | | | |
| **Vue.js** | **微信小程序** | **uni-app** | **React** |
| vue-router | pages.json /  wx.navigateTo( ..) | pages.json /  uni.navigateTo( ..) | 基于浏览器：react-router-dom  脱离浏览器：react-navigation |

**三、基于浏览器的React项目中的路由导航模块—— react-router-dom**

GitHub仓库：https://github.com/remix-run/react-router/blob/main/docs/getting-started/tutorial.md

英文官网：https://reactrouter.com/

使用方法：

①在当前项目中下载必需的模块文件

npm i react-router-dom

②使用方法： **老大爷**(路由器) >  **大脑**(路由词典/列表) > **线路**(路由\*N)

**BrowserRouter** > **Routes**  > **Route** \* N

<BrowserRouter>

<Routes>

<Route path="/地址" element={<元素名/> }/>

....

<Route path="/地址" element={<元素名/> }/>

<Route path="\*" element={<NotFound /> }/>

</Routes>

</BrowserRouter>

③路由跳转的两种方式：

模板法： <Link to="/地址">文字</Link>

脚本法： let nav = useNavigate(); nav('/地址')

**四、React中的列表渲染**

|  |  |  |
| --- | --- | --- |
| **Vue.js中的列表渲染** | **微信小程序中的列表渲染** | **React中的列表渲染** |
| v-for | wx:for |  |

**五、React应用中的异步数据请求**

|  |  |  |
| --- | --- | --- |
| **Vue.js中异步数据请求方案** | **微信小程序中异步数据请求方案** | **React中异步数据请求方案** |
|  |  |  |

课后任务：

①照猫画虎：完成“MfFooter组件”

②挑战性任务：继续完善“MfHeader”

③试着完成：“MfCarousel”—— 涉及到大量的生命周期方法