### 什么是react

react是一款前端框架,由facebook开发并与2013年开源,是目前最流行的前端框架,围绕react的相关技术有

\* react.js,用react开发网页相关

\* react native,用react开发应用

\* react vr,用react开发vr

### react 版本

\* react 16以后的版本也被叫做react fiber,目前react的最新版本是16.4

### react 的官方网站

https://reactjs.org/

### 安装前提

使用react前需要安装node环境

### 安装

\* 配置基本的webpack

> 注意,复制已有项目的配置时最好不要复制node\_modules,把package.json复制后,执行npm i

\* 安装react react-dom

npm install --save react react-dom

\* 安装 babel

npm i babel-core babel-loader babel-preset-env babel-preset-react --save-dev

\* 添加webpack配置loader

```

{

test:/\.js$/,

exclude: /(node\_modules)/,

use: {

loader: 'babel-loader',

options: {

presets: ['env', 'react']

}

}

}

```

### react组件

react组件其实就是页面中的一部分

### jsx语法

在js中使用html,遇到 HTML 标签（以 < 开头），就用 HTML 规则解析；遇到代码块（以 { 开头），就用 JavaScript 规则解析.

\* 注意点

> \* 在jsx语法中使用组件分为html组件和自定义组件,自定义组件必须大写字母开头

> \* 在返回组件内容时,如果不想有多余的标签,可以使用React.Fragment来代替html标签

\* 注释

{/\*注释\*/}

{

//注释

}

### 添加样式

\* 行内: style = {{color:'red'}}

### 绑定事件和this指向问题

< onClick = {this.handleClick.bind(this)} >

### this.state 存放组件内部数据

\* 初始化

constructor(props){

super(props);

this.state = {

}

}

\* 改变

this.setState({

attr:val

})

### 获取输入框数据的方法

\* 给输入框添绑定onChange事件

\* 在onChange的事件函数中通过event.target.value获取值,将获取到的值赋值给this.state

### 父组件给子组件传递参数

\* 父组件定义属性并赋值

\* 子组件通过this.props.属性名来接收

### 子组给父组件传递参数

子组件调用父组件传递过来的方法,将要传递的参数传给该方法

### react的特点

\* 声明式(数据驱动)

\* 组件化

\* 单向数据流

> 父组件可以给子组件传值,但子组件不能改变父组件的值

\* 视图层框架

\* 虚拟DOM

### PropTypes和DefaultProps

\* PropTypes 定义父组件传入子组件数据的校验规则

> 注意点:

>> \* 先引入,react会默认安装prop-types,所以无需安装

>> import PropTypes from 'prop-types';

```javascript

Item.propTypes = {

content:PropTypes.string,

index:PropTypes.number,

handleDelete:PropTypes.func,

test:PropTypes.string.isRequired

}

Item.defaultProps = {

test:'Hello'

}

```

\* DefaultProps 定义子组件中的默认值

\* https://reactjs.org/docs/typechecking-with-proptypes.html

### props state 和 render的关系

\* 当组件的state或者props发生改变时render函数会重新执行

\* 当父组件的render函数执行时,子组件的render函数也会被执行

### 虚拟DOM

虚拟DOM其实就是一个用来描述DOM节点的JS对象

1. state 数据

2. JSX 模板

3. 数据 + 模板 -> 真实DOM

4. 生成原始虚拟DOM

5. state 数据变化

6. 数据 + 模板 -> 新的虚拟DOM

7. 比较原始虚拟DOM和新的虚拟DOM,找出区别(diff算法)

8. 直接操作DOM改变有区别的DOM

### 获取DOM

\* 在dom元素上添加ref属性,例如:

ref = {(input)=>{this.input = input}}

> 注意点:由于setState是一个异步方法,如果需要获取最新的Dom,需要写在setState方法的第二个回调函数中

### 生命周期函数

在某个时刻组件会自动执行的函数

<img src="img/Lifecycle.png" alt="">

### 挂载Mounting(组件被第一次添加到页面上)

\* constructor(props)

\* static getDerivedStateFromProps(nextProps, prevState),多用于如果props有变化,需要更新state的场景,该方法返回state的更新

\* render()

\* componentDidMount(),组件挂载完毕执行,多用于发送ajax获取数据

### 更新Updating(state或者props发生改变)

\* static getDerivedStateFromProps(props, state)

\* shouldComponentUpdate(nextProps, nextState),该方法返回布尔值,根据返回的布尔值决定是否执行后续的周期函数

\* render()

\* getSnapshotBeforeUpdate(prevProps, prevState),该方法返回一个值,这个值会随后被传入到 componentDidUpdate 中使用

\* componentDidUpdate(prevProps, prevState,snapshot),组件更新完成后执行

### 卸载Unmounting(组件从页面中删除)

componentWillUnmount()

### 应用场景

\* shouldComponentUpdate用来阻止不必要的页面渲染

\* 在componentDidMount中发送ajax获取数据(axios)

\* 安装

npm install antd --save

\* 基本使用

import { DatePicker } from 'antd';

import 'antd/dist/antd.css';

\* 按需加载

> 安装babel-plugin-import

npm install --save-dev babel-plugin-import

> 修改webpack.confing.js文件的babel-loader option,添加

```

plugins: [

["import", { "libraryName": "antd", "libraryDirectory": "es", "style": "css" }]

]

```

### Redux概念

\* Redux是一个数据层框架

<img src="img/redux-concept.png" alt="">

### Redux工作流

<img src="img/redux-flow.jpg" alt="">

### 路由组成

\* 历史,以栈的形式记录页面访问过程

\* 跳转,不同页面之间跳转并可以传递参数

\* 事件,打开页面或是返回页面时触发的逻辑

### 路由分类

\* 页面路由(跳转时页面重新加载)

跳转:window.location.href = 'http://www.baidu.com'

回退:history.back()

\* hash路由(跳转时只有页面的hash值发生变化,页面不重新加载,根据hash值的变化决定页面的内容)

跳转: window.location.herf = '#hash'

事件: window.onhashchange = function(){

console.log(window.location.hash)

}

\* h5路由(h5 history对象提供操作路由的方法,可以通过向路由历史里添加新的路径并且页面不重新加载)

推进一个状态: history.pushState('name','title','/path')

替换一个状态: history.replaceState('name','title','/path')

事件:

window.onpopstate = function(e){

console.log(e.state)

console.log(window.location.href)

console.log(window.location.pathname)

console.log(window.location.hash)

console.log(window.location.search)

}

> onpopstate事件只处理后退前进

### 前端路由安装

npm i react-router-dom --save

### 基本使用

https://reacttraining.com/react-router

### React-Router

\* <BrowserRouter>,<HashRouter> 路由方式

\* <Route> 路由规则

\* <Switch> 路由选项

\* <Link>,<NavLink> 跳转,导航

\* <Redirect> 自动跳转

> 注意点:

>> \* Router组件是最外层组件

>> \* 使用BrowserRouter时,页面刷新会向服务器发送请求,而HashRouter不会

>> \* 使用BrowserRouter时devServer的historyApiFallback:true

### 自定义路由

\* 安装配置ES6扩展 babel

npm i babel-preset-stage-3 babel-preset-es2015 --save-dev

presets: ['env','es2015','react','stage-3'],

\* 定义

```javascript

const ProtectedRoute = ({ component: Component, ...rest }) => (

<Route {...rest} render={props => (

window.localStorage.getItem('userInfo') ? (

<Component {...props}/>

) : (

<Redirect to={{

pathname: '/login'

}}/>

)

)}

/>

)

```