Ssh-r1.ajax

一：创建XMLHTTPRequest对象

二：处理服务器上面返回的响应xml.onreadystatechange = function () {}

三：创建请求xml.open()

四：发送请求XML.send()

2.promiss

promise是异步编程的解决方案。有三种状态未完成（pending）、已完成（fulfilled）失败（rejected）

3.redux

是全局状态管理有三大原则一唯一数据源state、二：保持状态只读：在view层只能对state数据读取不能改变、三：数 据改变只能通过纯函数完成：reducer函数只做数据的运算不做数据的存储，接收两个参数：state是当前的数据状态， action是传入的值

redux的写法

首先要有一个制造商就是Provider他从react-redux里来他必须包裹着根节点<provider></provider>

import {Provider} from react-redux

然后创建store <provider store=createStore((state = {a:0})=> {})></provider>createStore里面必须传一 个对象其实里面就是reducer

import {createStore} from redux

然后到组件里连接组件connect()(Home)连接后得到dispatch从this.props解构出来dispatch里面是action写法是 dispatch({

type: 'ACTION\_HA',

a:10

})

improt {connect} from 'react-redux';

dispatch后走reducer写法是const reducer = (state={a:10}, action) => {

switch(state.type) {

case: 'ACTION\_HA',

return Object.assign({},state, {a:action.a})

}

}

走完之后到组件取数据connect(mapStateToProps, mapDispatchProps, mergeProps, options)()在他之前先定义

const mapStateToProps = (state) => {

a: state.a

}

4.redux-saga

他是redux的一个中间件主要集中处理架构中的异步处理工作他是rredux异步的解决方案，可以让redux里的 action 按顺序执行

写法：在store文件夹里行建一个sagas的文件夹里面有index.js和所需的js store.js里面也要改

import { createStore, applyMiddleware } from 'redux';

import thunkMiddleware from 'redux-thunk';

import {createLogger} from 'redux-logger';

import rootReducer from './reducer';

import rootSagas from './sagas';

import createSagaMiddleware from 'redux-saga';

const loggerMiddleware = createLogger()

const sagaMiddleware = createSagaMiddleware();

export default function configureStore (initialState) {

const store = createStore(

rootReducer,

initialState,

applyMiddleware(

thunkMiddleware,

sagaMiddleware,

loggerMiddleware

)

);

sagaMiddleware.run(rootSagas);

return store;

}

改完store.js后就可以写saga了

比如在saga里建一个home.js 里面写一些action什么的

import {takeEvery, takeLatest} from 'redux-saga';takeEvery是每点击一次执行一次操作、takeLatest是只 执行最后的点击事件

import {put, fork, call} from 'redux-saga/effects';put就相当于dispatch里面写actioncreater、fork是无 阻塞调用、call是用来请求数据的有两个参数第一个是promise对象,第二个是路径

function \* watcher () {yield takeEvery('REQUIRE\_HAHA', Worker);}

function \* worker () {try {

const data = yield call(axios.get, 'https://trad-pusher.8win.com/match/v2?k=cn010001');

const dataNew = JSON.parse(data.data.data);

yield put({

type: 'SUCCESS\_LIST',

dataNew

});

yield put({

type: 'SUCCESS111111'

});

} catch (error) {

yield put({

type: 'ERROR',

error: '出错了'

});

};}

在把worker 抛出

export default function \* root () {

yield fork(watcher);

};

在写saga里的index.js例如：

import root from './test';

import {fork} from 'redux-saga/effects';

import data from './home';

export default function \* rootSaga () {

yield fork(root);

yield fork(data);

};

5.reselect

是用来对state里传回来的值进行处理

写法：

在store文件夹里新建一个reselect.js

在组件里

const mapStateToProps = (state)=> {

return {

count: getHa(state)

}

}

上面的getHa方法就是从reselect.js里传来的

下面写reselect.js

import {createSelector} = 'reselect';

const gets (state) => { return state}

expore cost getHa = createSelector(

第一个参数是一个数组[gets].(arg) => {return arg}

)

6.axious和fetch 的区别：

Axios是一个基于promise的HTTP库，可以用在浏览器和node.js中

FetchAPI被定义在BOM的window对象中，你可以用它发起对远程资源的请求，改方法返回的是一个promse对象，让你能够对请求的返回结果进行检索。

7：Gitt和GitHub：

Git是版本管理工具

github是提供git的托管服务的 要使用github 必须先生成密钥，为git和GitHub的传输是通过ssh加密的 生成密钥： $ ssh-keygen -t rsa -C ["youremail@example.com"](mailto:\"youremail@example.com\")

Git 的步骤：1.git init 2.git add . 3.git commit -m ‘zhushi’4.git remote add origin https:// github.com/tianqing0204/mytry.git 5.git push -u origin master

分支： 首先先创建一个分支git branch dev(功能性分支) 在转到dev分支上： git checkout dev 一句代码表示：git checkout -b dev 然后你在dev分支上完成功能在到主分支上合并分支 git branch 查看分支 git checkout master 合并分支：git marge dev 合并完之后把次分支删除 git branch -d dev

8：微信小程序：

一、什么是微信小程序：

不需要下载安装即可使用的应用,成本低

二、微信小程序的特点：

适合做简单的、用完即走的应用

适合低频的应用

适合性能要求不高的应用

三、小程序对开发者的影响：

会提升市场对javascript程序员的需求

1. view相当于div/images/text是文本标签
2. 绑定事件时事件名前面都要加bind如（bindTab）也可以加catch他俩还是有点区别的用catch绑定事件会阻止事件冒泡。
3. 跳转页面是可以用redirectTo也可以用navigatoTo他俩的共同点是都接收一个对象作为参数，不同点是redirectTo跳转后是平级组件没有返回按钮，navigatoTo跳转后是子集组件，有返回按钮。
4. 用ait+shift+f可以快速的格式化代码。
5. template模板化的技术

9: WebSocket

HTML5提供的一种浏览器与服务器通讯的网络技术，属于应用层协议。

特点： WebSocket可以在浏览器里使用、支持双向通信、使用很简单、扩展性好

10:cookie:

cookie 是存储于访问者的计算机中的变量。每当同一台计算机通过浏览器请求某个页面时，就会发送这个 cookie。你可以使用 JavaScript 来创建和取回 cookie 的值。

11:web和app的通信:

用url scheme(他是一个协议有自定义协议比如：weixin://scanqrcode、mqq://scanqrcode、taobao://scanqrcode)

12:模板引擎的原理:

先获取html中对应的id下的innerhtml，利用开始标签和关闭标签进行字符串切分，其实是将模板划分成两部分内容，一部分是html部分，一部分是逻辑部分，通过区别一些特殊符号比如each、if 等来将字符串拼接成函数式的字符串，将将两部分各自经过处理后，再次拼接到一起，最后将拼接好的字符串采用new function()的方式转化成所需要的函数。

（把模板引擎分为两个部分，一个是html部分，一个是逻辑部分，把这两部分经过一些处理后拼接到一起，把拼接好的字符串采用new function 的方式转化成所需要的函数。）

模板引擎的好处：

实现界面与数据的分离，提升开发效率，使代码重用变得更加容易。

13:webpack4: 一：npm install webpack@nextwebpack-cli --save-dev

二：性能提升，webpack会自动使用UglifyJs插件，放弃了Node v4的支持以大量支持ES6语言

三：会自动找你的entry

四：去除了CommonsChunkPlugin插件

14:什么是mvvm: 从作用域中取得scope对象的属性，（M）,绑定到页面当中（V）， 页面的值发生变化（V）会反应到scope对象上（M）.

15:一个页面从输入url到页面加载显示完成，这个过程发生了什么？

1. 当发送一个url请求时，浏览器会开启一个线程处理这个请求， 同时在远程DNS服务器上启动一个DNS查询。使浏览器获得请 求对应的ip地址。
2. 浏览器与远程web服务器通过TCP三次握手协商来建立一个TCP/IP连接。
3. 一旦TCP/IP连接建立，浏览器会通过该连接向远程服务器发送HTTP的GET请求。远程服务器找到资源并使用HTTP响应返回改资源，值为200的HTTP响应状态表示一个正确的响应。
4. WEB服务器提供资源服务，客户端开始下载资源。

16：windows Android环境搭建：

一、

17： 安装node：

先找到包后下载安装，在cmd里输入npm config set registry <https://registry.npm>.taobao.org -g

npm config set disturl [https://npm](https://registry.npm).taobao.org/dist -g

18: 作用域链和原型链：

作用域链：每一个函数都有一个作用域，如果一个函数内部又嵌套另一个函数，嵌套的函数也会有一个作用域，嵌套函数可以访问外部函数的变量以及全局作用域的变量，此时就形成了一条作用域链。（他和闭包的区别在于嵌套函数可以访问函数外部的变量但不可以访问全局作用域的变量）

原型链：每个对象都有一个\_proto指针，每个构造函数都有一个prototype属性，对象是有原型对象的，原型对象也有原型对象，对象的原型对象一直往上找，会找到一个null，这样就形成了原型链。

19： instanceof

测试一个对象在其原型链中是否存在一个构造函数的prototype属性

20：怎样判断用户是否登录状态：

用户自动登录后把用户名或者唯一标识用户的字符串在服务器用密钥加密，然后和未加密的字符串一起保存在cookie中，相当于给浏览器发了一个可以不提供用户密码就登录的认证。当收到一个没有session的连接时，就检验cookie里的标识符和加密后的字符串是否匹配，是的话就说明这是本服务器提供的登录认证，然后就可以根据那个唯一标识符从数据库里提取用户信息放入session中了。

21：在进行技术选型时应该注意什么：

1. 选择最熟悉的技术
2. 选择拥有强大的社区支撑的开源技术
3. 确保技术前进步伐
4. 学会从业务端开始思考
5. 先验证，后使用
6. 重视经验

22：如何提升渲染列表页的速度：

使用懒加载，使用首屏加载，模板引擎,服务端渲染

23：模板引擎：

ejs jade juicer artTemplate handlebars

24:Amd和Cmd

是JS的模块化，sea.js遵循的是cmd规范，对于依赖的模块是依赖就近。延迟执行，require遵循的是AMD的规范，对于依赖的模块是依赖前置，提前执行。

25:上传图片的逻辑：

写一个input 框

<input type='file' onChange={this.upLoad.bind(this)} />

利用onchange属性绑定事件

通过e.target.files判断图片的大小以及类型

实例化一个FormData对象存放图片的路径

发送一个post请求吧图片的路径发送过去

创建文件读取对象

设置上传图片的数据

用react提供的createElement方法创建一个标签存放图片

后台创建一个上传图片的接口

引入multer，multer是nodeJS文件上传的中间件。

设置保存路径

*const* upload = multer({dest: './public/img'}).any();

就可以了。

26:一些数组的方法：

1. 生成六位数字的验证码：

Math.random().toString().slice(-6);

Math.random().toFixed(6).slice(-6);

1. n维数组转一维数组：

Var foo =[1,[2,3],[‘4’, 5,[‘6’,7,[8]]],[9],10];

Foo.tostring().split(‘,’);

1. 统计字符串中相同字符串出现的次数：

Var arr = ‘abcdaabc’;

Var info = arr.split(‘’).reduce((p, k) => (p[k]++ || (p[k]= 1), p), {});

Console.log(info);

27:组件化和模块化的区别：

组件化更多关注的是UI层，模块化更多的是功能和数据的封装。

28：JS的设计模式有哪些：

单体模式，确保一个类只有一个实例，使用场景：用于创建独一无二的只有一个实例的对象。

观察者模式（发布订阅模式）

工厂模式例如面向对象

混合模式

常用的是混合模式因为他高复用，代码量少

如果要把一些方法封装起来的话就用面向对象，可以高复用低耦合

29：Vue实现数据双向绑定的原理：

通过object.definpropty来劫持各个属性setter和getter，在数据改变是发布消息给订阅者，触发相应的监听回调。

vue的双向邦定是基于ES5 中的 getter/setters来实 现的，而angular而是由自己实现一套模版编译规则，需 要进行所谓的“脏”检查，vue则不需要。因此，vue在性 能上更高效，但是代价是对于ie9以下的浏览器无法支持。

30：用node起服务前端刷新路由消失了：

使用哈希路由

31：三份webpack:

Webpack.dev.config

Webpack.base.config

Webpack.produce.config

生产环境就是线上环境不需要自己起服务需要压缩打包index.html和js.

开发环境需要自己模拟接口dev

32: 跨域 ：

33： component和methods的区别

Computed：有缓存，性能更高，会自动检测

Methods：用来定义方法，需要调用

34：h5和app的通信：

通过jsbridge调用一个window的全局方法,拿到回调函数里的接口，h5页面通过jsBridge对象触发URLscheme调用这些接口。

35：如何与后端进行接口对接：

领导带我们开会，对接需要什么字段和接口

36：前端安全：

xss攻击、会话劫持、 SQL注入

如何防止：md5加密、下载一个xss插件

37:webpack里写代理：

写一个setup方法传一个app方法里面写app.get(‘接口’，回调 )

Setup(app) {

App.get(‘/api/xzq’, (req, res) => {

Const list = require(‘./src/mock/data/list.json’);

Res.json(list);

})

}

38：Vue组件间的通信：

父向字用props、自向父用自定义事件在子组件用v-on接收、兄弟组件用bus、还可以用vuex 用$parent可以获取父组件的实例、用$children可以获取子组件的实例、用$root可以获取组件所在的根实例、用this.refs可以获取子组件。

39:react组件的通信：

父向子用props、子向父通过props调用父组件里传过来的自定义事件、兄弟组件用EventEmitter这个插件、子孙级别的组件：在孙子组件获取context上下文的状态，在爷爷组件里getChildContext方法，获取子组件的状态。全局通信用redux.

40:分页的思路：

41:react里的state和props的区别：

Prop:定义外部的接口，如果外部组件需要改变数据的时候用，组件内部不能更改。

State:可以改变组件内部的数据

42：移动端的兼容问题：

华为手机和小米手机的某些机型，有虚拟键盘，需要把图缩小

安卓手机需要开localstro苹果手机不需要开

43：响应式的项目：

媒体查询

44：懒加载和预加载：

懒加载就是按需加载，预加载就是滑到第一屏就加载第二品和第三屏了

45：图片懒加载的原理：

通过判断滑到的距离大于图片的高度

46:继承：

1. 原型链继承：将父类的实例作为字类的原型，无法实现多继承，无法向父类的构造函数穿参
2. 构造继承：复制父类的实例属性给子类
3. 组合继承：调用父类的构造，实现父类属性的继承，将父类的实例作为子类的原型，生成了两份实例。
4. 寄生组合继承：继承父类的构造函数，创建一个没有实例方法的类

47:MVC和MVVM的区别：

48：跨域的几种方式：

1. jsonp利用动态创建一个script标签利用src属性需要传入路径提供一个callback来接收数据。浏览器会调用callback解析json对象。
2. 通过修改document.domain来跨域，他的局限性就是一级域名一致才可以跨域。两个document.domain相同就可以跨域了。
3. Windown.name:动态创建iframe,动态注册各种事件，获取完数据后在把iframe销毁。
4. Window.postMessage:是h5的新特性，可以使用它向其他window对象发送数据

49：gulp，grunt、webpack区别：

50：mvc、mvvm、mv\*、mvp的区别 ：

1. MVC：用户看见的是view（HTML）层，通过一些点击事件或输入一些数据做一些交互，视图层会把这些操作传递给控制器，控制器会根据用户的操作查询和修改底层数据，底层数据的改变会与服务器做一些交互以获取或更新数据信息，底层数据的改变会根据数据绑定机制传递给view层。
2. Mvvm:视图数据的变化同时会修改数据源，数据源数据的变化会反应到视图层。核心是数据双向绑定思想。
3. MVP:把MVC中的控制器换成了presenter呈现。是为了切断view于moduel的联系用presender做桥梁

51：依赖注入：

是一种软件设计模式，可以减少组件间的耦合

52：创建函数的几种方式：

1. 函数声明：function sum1(num1, mun2) {}
2. 函数表达式：var = function (){}
3. Var sum3 = new function(){}

53:前端的性能优化：

1. 减少http的请求，因为请求100kb图片比下载两个50kb的图片要快，每个请求都会对服务器和浏览器产生性能负担，应该合并图片，合并css和js文件
2. 减少对dom的操作，因为修改dom会造成页面的重汇和重排，更不能对dom循环，
3. 使用json格式来进行数据交换，因为json是轻量级的数据交换格式，在json中有两种结构对象和数组

54：前端的安全问题：

1. XSS跨站脚本攻击，本质原因在于，浏览器错误的将攻击者提供的用户输入数据当做js脚本给执行了，最佳的做法就是对数据进行严格的输出编码。
2. 通过iframe在页面上添加第三方提供的广告、天气预报、社交分享插件等，iframe的内容是由第三方来提供的，默认情况下是不受我们控制的。在iframe标签上添加sandbox属性对iframe的行为进行各种限制。

55: 移动端的单位：

绝对单位是px；相对单位是em

em是相对于body的单位，有继承性，默认的是16px,所以想设置10px为1rem 的话就给body设置font-size:62.5%;

rem是相对于html没有继承性，

我最近做的项目是本来生活网，主要使用的技术是react，在首页一进去需要调用必读题图api获取当地的地址，右侧有一个登陆的按钮点击这个按钮调到登陆页面点当用户输入用户名和密码的时候点击登录的按钮会把用户名在服务器上加密和未加密的用户名一起存到cookie中当在一次登录的时候会把加密的用户名和未加密的用户名和未加密的用户名做对比如果一样的话会自动登录首页顶部有一个导航栏当点击这个导航的时候会改变不同的数据这个功能使用里的tab插件利用这个插件的onchange时间获取不同的ID改变数据，首页还有一些商品列表这些商品列表是按需加载的使用pulltorefres插件实现的点击商品列表进入商品列表页面然后就可以通过location的仓鼠利用这个参数请求响应的数据

8.做项目遇到的问题：

1：如果把app.js 用connect连接后路由就无法跳转了。

解决方法： 一：不要平常connect连接组件直接在class上边写@connect （mapStateToProps）在把mapStateToProps这个方法放到前边（connect是高阶函数 所以可 以这样写） 二： 在connect(mapStateToProps, undefied, undefied, {purd:false}) (Home)

2: 在项目中请求大量的数据并且要切换页面的时候图片加载不出来

解决方法： 我们把这个请求写成两种状态一种是请求时一种是请求成功，在请求时把他的数据为空就可以解决 这个问题写两个action操作这个数据请求中的action数据情况成功的action传入相应的数据

3： 用源生写tab切换数据的时候有鼠标的滑动事件，当鼠标滑动到一定程度发送请求，请求数据但是有个问题会发送 多个请求

解决方法： 我们可以在constrol里面设置一个变量锁，让他发送一次请求成功后把变量设置为false滑动到一 定程度在设置为true