**1.对React很理解**

答：

react起源于Facebook的内部项目，区别于其他MVC框架，我更认为是MVC中的V，本身react（Facebook）不认可MVC框架因此才自己设计react.相较于其他热门框架，react是函数式编程框架，把html甚至css都可以嵌套进js中，形成自己独特的语言jsx.react的思想是一切皆组件，把所有模块不管大小都可以切割成一个个小组件，组件可组合，可服用，可维护（每个组件仅包含自身逻辑），只要管理好状态，状态通过react框架就能管理好每一个UI组件。

react的核心是引入了虚拟DOM（Virtual DOM）的机制,在react中会创建一个虚拟DOM，浏览器的DOM构造是基于虚拟DOM进行的，每当数据变化时，react会重新构建一个虚拟DOM，拿它跟上一个虚拟DOM进行对比，得到DOM结构的区别，仅仅将这部分需要变化进行浏览器的DOM跟新，这也是react性能的更优的其中一个原因。因为虚拟DOM是内存数据，性能极高。

**2.对Vue的理解**

答：

vue作为后起之秀，同时吸收了angular和react的优点的框架，我个人认为它是一款轻量级，性能优秀，简单易用的框架，并且因为是中国人开发，中文文档写的也更详细，更容易入门了，对于我而言，如果项目并不复杂，希望快速开发简单易用的框架，那么vue是不错的选择，

谈回框架本身，首先他可以说是MVC框架也可以是MVVM的框架，拥有双向数据绑定的能力，是以数据驱动的形式来操作DOM，我们只需要关注model层，view交由vue去处理。

其次vue也使用了Virtual DOM，因此拥有不错的性能。...

组件化开发，也是vue所提倡的，不同于jsx，vue的组件会提倡html/css/js分离，在vue组件中，templete是视图层是html结构，script是model层。关注于数据等相关逻辑，style是样式层，关注于组件本身的样式，同样的vue的组件也是由可复用，可组合，可维护的特点。

**3.对面向对象很理解**

答：

面向对象编程，OOP是一种变成规范，它有封装，继承，多态等特点，简单的讲，就是将需要的属性，或者函数方法封装到对象里，需要使用调用对象的属性或者方法来调用，上升一个层面，我们可以把特定功能的属性，方法封装在一起，分文别类，方便复用，传参；此外通过继承可以快速的将对象的属性方法继承到另一个对象，不用重新定义申明。

多态的基本概念：一个引用类型（变量）在不同情况下的多种状态。

同一操作作用于**不同的对象**，可以有不同的解释，产生不同的执行结果。

**5.你的项目，**Redux怎么分层

答：

view /store/action/reducers/middleware/

**6.socket断开后，如何重连**

socket 的理解，TCP协议的一种实现方式，根HTTP的区别，长连接短链接。 websocket是TCP的一种实现方式。

浏览器不会执行websocket 的 onclose方法，我们无法知道是否断开连接，也就无法进行重连操作。

定时发送websocket数据到后端（心跳检测），一旦请求超时，onclose便会执行，这时候便可进行绑定好的重连操作。(reconnect)

因此websocket心跳重连就应运而生。

如果希望websocket连接一直保持，我们会在close或者error上绑定重新连接方法。

ReconnectingWebSocket 是一个小型的 JavaScript 库，封装了 WebSocket API 提供了在连接断开时自动重连的机制。

//间隔发送心跳包数据给服务器，服务器在一定时间内发回心跳包响应，对比超时限定，如果超过设定的超时时间，则认为当前与服务器的websocket连接已经断开，关闭当前web socket连接，善后处理，例如重新连接，或者弹出提示……

**7.如何实现vue无限级树型导航(2种及以上方法)**

答：1用递归的方式把所有Dom节点遍历完之后在v-html插入template

2.绑定指令，先渲染一级菜单，设定点击事件，判断该菜单是否由子级菜单，由的展示并且再次绑定同样的事件，没有不显示

**8.MVVM的理解，MVC的理解**

答：是两种设计模式 ，但vue即可以是MVC由是MVVM，M是DATA,V是tempele, C是method 生命周期等。

**10.嵌套路由和嵌套组件是否由区别**：没有半毛钱区别，一样的

**11.表格封装是思路需要的注意思想：**

**12.轻量级**：只专注实现核心双向绑定功能，其他没有，angular包括HTTP，等很多功能，通过判断功能的多于少判断轻量级

**13.v-model**只能用于表单元素，js操作属性是不能实现v-model,.v-model 的底层原理，js属性监听，用js改变input框的数值，能不能触发v-model，答：不能。

**14.jQuery,的优点。**最强大的选择器！ 链式调用！

**15.vuex 组件A和组件B，组件A由渲染，B没有， 问题 组件A能不能拿到组件B的数据**

答：能，vuex中A能通过dispath()调用B里面的方法

**16.vuex需要注意什么。**答：属性名称不要重复

**17继承和原型链 继承的优点，**复用父类属性方法，子类可重新在定义以及扩展，另外一点是可以把父类整个原型链继承过来，包括父类的父类一直往上。

**18.vue data和method里面的this指向是否一样**

答：

vue中把方法写在data和method里是一样的可以绑定事件调用，但由点区别的是this指向并不相同，data里面的方法this指向是全局代理对象Proxy，methods里的方法中this指向组件component本身。

**19.post收不到数据，**

答： 如果不是fromdata 而是object是收不到，如果是formdata才能发送到后端（或者后端设置，反正必须对应。一般也是用formdata）

POST表单请求提交时，使用的Content-Type是application/x-www-form-urlencoded，而使用原生AJAX的POST请求如果不指定请求头RequestHeader，默认使用的Content-Type是text/plain;charset=UTF-8。

**20.vue-resource和拦截器**

答：vue-resource是一个非常轻量的用于处理HTTP请求的插件，它提供了两种方式来处理HTTP请求：

* 使用Vue.http或this.$http ( 常用this.$http.get(url,option).then(success,error) )
* 使用Vue.resource或this.$resource (服务)

支持拦截器(inteceptor)

拦截器是全局的，拦截器可以在请求发送前和发送请求后做一些处理。

拦截器在一些场景下会非常有用，比如请求发送前在headers中设置access\_token，或者在请求失败时，提供共通的处理方式。

Vue.http.interceptors.push((request, next) => {

// ...

// 请求发送前的处理逻辑

// ...

next((response) => {

// ...

// 请求发送后的处理逻辑

// ...

// 根据请求的状态，response参数会返回给successCallback或errorCallback

return response

})

})

运用场景:请求发送前显示loading，接收响应后隐藏loading。

**21.vue-route 路由拦截**

定义路由的时候就需要多添加一个自定义字段requireAuth，用于判断该路由的访问是否需要登录。如果用户已经登录，则顺利进入路由，

否则就进入登录页面。

{

path: '/repository',

name: 'repository',

meta: {

requireAuth: true, // 添加该字段，表示进入这个路由是需要登录的

},

component: Repository

},

利用beforeEach()钩子函数进行判断

router.beforeEach((to, from, next) => {

if (to.meta.requireAuth) { // 判断该路由是否需要登录权限

if (store.state.token) { // 通过vuex state获取当前的token是否存在

next();

}

else {

next({

path: '/login',

query: {redirect: to.fullPath} // 将跳转的路由path作为参数，登录成功后跳转到该路由

})

}

}

else {

next();

}

})

**22.route的底层实现原理**

hash模式的路由,是通过监听hashchange来实现

window.addEventListener('hashchange', () => { // this.transitionTo(...) })

* new Vue()
* 执行 vue-router 注入的 beforeCreate 钩子函数
* 执行 router.init(vm)
* 执行 history.setupListeners()，注册事件监听

<router-view> 作为根组件下的子组件，从根组件那里可以获取到当前的路由对象，进而根据路由对象的组件配置，取出组件并用其替换当前位置的内容。这样，也就完成整个路由变更到视图变更的过程。

路由变更到视图变更的过程整理为：

hashchange --> match route --> set vm.\_route --> <router-view> render() --> render matched component

实现过程中的技术点包括：

* Vue 插件机制
* 响应式数据机制
* Vue 渲染机制
* 地址变更监听

**23.数据驱动的原理**

把一个普通对象传给 Vue 实例作为它的 data 选项，Vue.js 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。这是 ES5 特性，不能打补丁实现，这便是为什么 Vue.js 不支持 IE8 及更低版本。

用户看不到 getter/setters，但是在内部它们让 Vue.js 追踪依赖，在属性被访问和修改时通知变化。一个问题是在浏览器控制台打印数据对象时 getter/setter 的格式化不同，使用 vm.$log() 实例方法可以得到更友好的输出。

模板中每个指令/数据绑定都有一个对应的 watcher 对象，在计算过程中它把属性记录为依赖。之后当依赖的 setter 被调用时，会触发 watcher 重新计算 ，也就会导致它的关联指令更新 DOM。

**24.{{}}插值法的实现原理**

vue采用的是使用“Mustache”语法 ,以双花括号为标识,匹配到有双{{}}是会自动替换成键值

{{}}就是 Mustache 的标示符，花括号里的 data 表示键名，这句的作用是直接输出与键名匹配的键值

**25.cordova**

Cordova是一个用基于HTML、CSS和JavaScript的，用于创建跨平台移动应用程序的快速开发平台。它使开发者能够利用iPhone、Android、Palm、Symbian、WP7、Bada和Blackberry等智能手机的核心功能——包括地理定位、加速器、联系人、声音和振动等，此外Cordova拥有丰富的插件，可以调用。

安装 npm install cordova

创建App cordova create CordovaProject cn.dk.firstcordova CordovaApp

添加平台 cordova platform add android

构建和运行 cordova build android

**26.canvas**

<canvas> 标签只是图形容器，您必须使用脚本来绘制图形。

你可以通过多种方法使用Canva绘制路径,盒、圆、字符以及添加图像

获取canvas后

创建对象 var ctx=c.getContext("2d");

fillStyle属性设置css样式

fillRect(*x,y,width,height*) 方法定义了矩形当前的填充方式。

画线:

* moveTo(*x,y*) 定义线条开始坐标
* lineTo(*x,y*) 定义线条结束坐标

圆形:

* arc(x,y,r,start,stop)

绘制文本:

* font - 定义字体
* fillText(*text,x,y*) - 在 canvas 上绘制实心的文本
* strokeText(*text,x,y*) - 在 canvas 上绘制空心的文本

渐变:

* createLinearGradient(*x,y,x1,y1*) - 创建线条渐变
* createRadialGradient(*x,y,r,x1,y1,r1*) - 创建一个径向/圆渐变

图像:

* drawImage(*image,x,y*)

**27.mangodb的了解**

MongoDB 是一个基于分布式文件存储的数据库.MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

**28.mangodb和mysql的对比**

MySQL

关系型数据库。

在不同的引擎上有不同 的存储方式。

查询语句是使用传统的sql语句，拥有较为成熟的体系，成熟度很高。

开源数据库的份额在不断增加，mysql的份额页在持续增长。

缺点就是在海量数据处理的时候效率会显著变慢。

Mongodb

非关系型数据库(nosql ),属于文档型数据库。先解释一下文档的数据库，即可以存放xml、json、bson类型系那个的数据。这些数据具备自述性（self-describing），呈现分层的树状数据结构。数据结构由键值(key=>value)对组成。

存储方式：虚拟内存+持久化。

查询语句：是独特的Mongodb的查询方式。

适合场景：事件的记录，内容管理或者博客平台等等。

架构特点：可以通过副本集，以及分片来实现高可用。

数据处理：数据是存储在硬盘上的，只不过需要经常读取的数据会被加载到内存中，将数据存储在物理内存中，从而达到高速读写。

成熟度与广泛度：新兴数据库，成熟度较低，Nosql数据库中最为接近关系型数据库，比较完善的DB之一，适用人群不断在增长。

优势：

* 快速！在适量级的内存的Mongodb的性能是非常迅速的，它将热数据存储在物理内存中，使得热数据的读写变得十分快，
* 高扩展！
* 自身的Failover机制！
* json的存储格式！

缺点：主要是无事物机制！

分析一下Mysql和Mongodb应用场景

* 1.如果需要将mongodb作为后端db来代替mysql使用，即这里mysql与mongodb 属于平行级别，那么，这样的使用可能有以下几种情况的考量： (1)mongodb所负责部分以文档形式存储，能够有较好的代码亲和性，json格式的直接写入方便。(如日志之类) (2)从data models设计阶段就将原子性考虑于其中，无需事务之类的辅助。开发用如nodejs之类的语言来进行开发，对开发比较方便。 (3)mongodb本身的failover机制，无需使用如MHA之类的方式实现。
* 2.将mongodb作为类似redis ，memcache来做缓存db，为mysql提供服务，或是后端日志收集分析。 考虑到mongodb属于nosql型数据库，sql语句与数据结构不如mysql那么亲和 ，也会有很多时候将mongodb做为辅助mysql而使用的类redis memcache 之类的缓存db来使用。 亦或是仅作日志收集分析。

**30. swiper插件**

加载js和css

根据swiper写需要的html标签 例如: <**div** class="swiper-scrollbar"></**div**>

根据类名写css样式

初始化swiper

**var** mySwiper = **new** Swiper ('.swiper-container', { direction: 'vertical', loop: true, // 如果需要滚动条 scrollbar: '.swiper-scrollbar', })

**31.vue通信的几种方法**

props父传子

触发事件子传父

new一个vue创建事件中心触发事件 非父子

$ref :给子组件标签上写上标记, 父组件通过this.$refs.标记 可以得到子组件的实例对象

$children 得到数组,是所有子组件实例对象的集合

$parent 得到父组件实例对象

$root 得到跟组件实例对象,如果当前实例没有父实例,那么将返回自己(自己就是根实例了)

**32.前端跨域解决方法**

jsonp

cors 跨域资源共享

服务器代理

Cross-Origin Resource Sharing（CORS）跨域资源共享是一份浏览器技术的规范，提供了 Web 服务从不同域传来沙盒脚本的方法，以避开浏览器的同源策略，是 JSONP 模式的现代版。与 JSONP 不同，CORS 除了 GET 要求方法以外也支持其他的 HTTP 要求。用 CORS 可以让网页设计师用一般的 XMLHttpRequest，这种方式的错误处理比 JSONP 要来的好。另一方面，JSONP 可以在不支持 CORS 的老旧浏览器上运作。现代的浏览器都支持 CORS。

**CORS 与 JSONP 的对比**

CORS 除了 GET 方法外，也支持其它的 HTTP 请求方法如 POST、 PUT 等。

CORS 可以使用 XmlHttpRequest 进行传输，所以它的错误处理方式比 JSONP 好。

JSONP 可以在不支持 CORS 的老旧浏览器上运作。

vue页面缓存，混合，遇到的坑，前端优化手段，webpack设置代理，

33.vue 混合

可以设定一个混合对象,在每个组件中添加了这个混合对象就有了对于的属性和方法,如果后续再添加了重名属性,则会替换,混合对象可以给多个组件复用

34.vue的生命周期

* [beforeCreate](https://cn.vuejs.org/v2/api/#beforeCreate)
* [created](https://cn.vuejs.org/v2/api/#created)
* [beforeMount](https://cn.vuejs.org/v2/api/#beforeMount)
* [mounted](https://cn.vuejs.org/v2/api/#mounted)
* [beforeUpdate](https://cn.vuejs.org/v2/api/#beforeUpdate)
* [updated](https://cn.vuejs.org/v2/api/#updated)
* [activated](https://cn.vuejs.org/v2/api/#activated)
* [deactivated](https://cn.vuejs.org/v2/api/#deactivated)
* [beforeDestroy](https://cn.vuejs.org/v2/api/#beforeDestroy)
* [destroyed](https://cn.vuejs.org/v2/api/#destroyed)
* [errorCaptured](https://cn.vuejs.org/v2/api/#errorCaptured)

35.前端的优化

1.减少http次数 缓存设置,资源合并(css,js合并压缩) ,精灵图,懒加载 js置于底部或者异步加载.避免重复资源请求

2.减少DOM操作

3.vue组件优化.不需要首屏加载的组件可以异步加载

4.打包 vender 时不打包 vue、vuex、vue-router、axios 等，换用国内的 [bootcdn](http://www.bootcdn.cn/) 直接引入到根目录的 index.html 中。

36.vue页面缓存

keep-alive

37.watch 监听不到变化

一个是对象需要深度监听

数组改变没有用$set方法也监听不到

还有就是不是初始化的数据,是后续set增加的也监听不到

38.vue的坑

关于ref注册时间的重要说明: 因为ref本身是作为渲染结果被创建的，在初始渲染的时候你不能访问它们 - 它们还不存在！**$refs 也不是响应式的**，因此你不应该试图用它在模版中做数据绑定。

在父组件componets中直接注册子组件的时候里面的this是子组件实例而不是父组件的