1.es6 的新特性都有哪些？
let 定义块级作用域变量
没有变量的提升，必须先声明后使用
let 声明的变量，不能与前面的 let，var，conset 声明的变量重名
const 定义只读变量
const 声明变量的同时必须赋值，const 声明的变量必须初始化，一旦初始化完毕就不允许修改
const 声明变量也是一个块级作用域变量
const 声明的变量没有“变量的提升”，必须先声明后使用
const 声明的变量不能与前面的 let， var ， const 声明的变量重
const 定义的对象\数组中的属性值可以修改,基础数据类型不可以
ES6 可以给形参函数设置默认值
在数组之前加上三个点（...）展开运算符
数组的解构赋值、对象的解构赋值
箭头函数的特点
箭头函数相当于匿名函数，是不能作为构造函数的，不能被 new
箭头函数没有 arguments 实参集合,取而代之用...剩余运算符解决
箭头函数没有自己的 this。他的 this 是继承当前上下文中的 this
箭头函数没有函数原型
箭头函数不能当做 Generator 函数，不能使用 yield 关键字
不能使用 call、apply、bind 改变箭头函数中 this 指向
Set 数据结构，数组去重

2.常见的设计模式有哪些
1、js 工厂模式
2、js 构造函数模式
3、js 原型模式
4、构造函数+原型的 js 混合模式

5、构造函数+原型的动态原型模式
6、观察者模式
7、发布订阅模式

3.js 继承方式有哪些？
原型链继承
核心： 将父类的实例作为子类的原型
构造继承
核心：使用父类的构造函数来增强子类实例，等于是复制父类的实例属性给子类
实例继承
核心：为父类实例添加新特性，作为子类实例返回
拷贝继承
组合继承
核心：通过调用父类构造，继承父类的属性并保留传参的优点，然后通过将父类实例作为
子类原型，实现 函数复用
寄生组合继承
核心：通过寄生方式，砍掉父类的实例属性，这样，在调用两次父类的构造的时候，就不
会初始化两次实 例方法/属性，避免的组合继承的缺点

4.从输入 url 地址到页面相应都发生了什么？
1、浏览器的地址栏输入 URL 并按下回车。
2、浏览器查找当前 URL 是否存在缓存，并比较缓存是否过期。
3、DNS 解析 URL 对应的 IP。
4、根据 IP 建立 TCP 连接（三次握手）。
5、HTTP 发起请求。
6、服务器处理请求，浏览器接收 HTTP 响应。
7、渲染页面，构建 DOM 树。
8、关闭 TCP 连接（四次挥手）

5.session、localStorage 的区别
在较高版本的浏览器中，js 提供了 sessionStorage 和 globalStorage。在 HTML5 中提供了 localStorage
来取代 globalStorage。
html5 中的 Web Storage 包括了两种存储方式：sessionStorage 和 localStorage。
sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访
问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别
的存储。

而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

6.js 中跨域方法
1、jsonp 跨域(只能解决 get）
原理：动态创建一个 script 标签。利用 script 标签的 src 属性不受同源策略限制，因为所有的 src 属性和 href 属性都不受同源策略的限制，可以请求第三方服务器资源内容
步骤：1.去创建一个 script 标签
2.script 的 src 属性设置接口地址
3.接口参数，必须要带一个自定义函数名，要不然后台无法返回数据
4.通过定义函数名去接受返回的数据

2、document.domain 基础域名相同 子域名不同
3、window.name 利用在一个浏览器窗口内，载入所有的域名都是共享一个 window.name
4、服务器设置对 CORS 的支持
原理：服务器设置 Access-Control-Allow-Origin HTTP 响

应头之后，浏览器将会允许跨域请求

5、利用 h5 新特性 window.postMessage()

7.前端有哪些页面优化方法?

减少 HTTP 请求数

从设计实现层面简化页面

合理设置 HTTP 缓存

资源合并与压缩

合并 CSS 图片，减少请求数的又一个好办法。

将外部脚本置底（将脚本内容在页面信息内容加载后再加载）

多图片网页使用图片懒加载。

在 js 中尽量减少闭包的使用

尽量合并 css 和 js 文件

尽量使用字体图标或者 SVG 图标，来代替传统的 PNG 等格式的图片

减少对 DOM 的操作

在 JS 中避免“嵌套循环”和 “死循环”

尽可能使用事件委托（事件代理）来处理事件绑定的操作

8.数组去重的方法

ES6 的 set 对象

先将原数组排序，在与相邻的进行比较，如果不同则存入新数组

```
function unique(arr){
var arr2 = arr.sort();
var res = [arr2[0]];
for(var i=1;i<arr2.length;i++){
if(arr2[i] !== res[res.length-1]){
res.push(arr2[i]);
}
}
return res;
}
```

利用下标查询

```
function unique(arr){
var newArr = [arr[0]];
for(var i=1;i<arr.length;i++){
if(newArr.indexOf(arr[i]) == -1){
newArr.push(arr[i]);
}
}
return newArr;
}
```

9.ajax 中 get 和 post 请求的区别

get 一般用于获取数据

get 请求如果需要传递参数，那么会默认将参数拼接到 url 的后面；然后发送给服务器；

get 请求传递参数大小是有限制的；是浏览器的地址栏有大小限制；

get 安全性较低

get 一般会走缓存，为了防止走缓存，给 url 后面每次拼的参数不同；放在?后面，一般用个时间戳

post 一般用于发送数据

post 传递参数，需要把参数放进请求体中，发送给服务器；

post 请求参数放进了请求体中，对大小没有要求；

post 安全性比较高；

post 请求不会走缓存；

10.ajax 的状态码 400 错误请求服务器不理解的语法

2 开头

200 : 代表请求成功；

3 开头

301 : 永久重定向；

302: 临时转移

304 : 读取缓存 [表示浏览器端有缓存，并且服务端未更新，不再

向服务端请求资源]
307:临时重定向
以 4 开头的都是客户端的问题;
400 :错误请求服务器不理解的语法
401: 权限不够;（身份不合格，访问网站的时候，登录和不登录是不一样的）
404 : 路径错误，找不到文件
以 5 开头都是服务端的问题
500 : 服务器的问题
503: 超负荷;

11.Vuex 中的属性
state 中保存着共有数据，数据是响应式的
getter 可以对 state 进行计算操作，主要用来过滤一些数据，可以在多组件之间复用
mutations 定义的方法动态修改 state 中的数据，通过 commit 提交方法，方法必须是同步的
actions 将 mutations 里面处理数据的方法变成异步的，就是异步操作数据，通 store.dispatch 来
分发 actions，把异步的方法写在 actions 中，通过 commit 提交 mutations，进行修改数据。
modules：模块化 vuex

12.DOM diff 原理
1.把树形结构按照层级分解，只比较同级元素。
2.给列表结构的每个单元添加唯一的 key 属性，方便比较。
3.React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
4.合并操作,调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束,
5.查 React 检查所有标记 dirty 的 component 重新绘制.
6. 选择性子树渲染。开发人员可以重写 shouldComponentUpdate

提高 diff 的性能。

13.Promise A+规范
他是 ES6 中新增加的一个类（new Promise）,目的是为了管理 JS 中的异步编程的，所以把他称为
“Promise 设计模式”
new Promise 经历三个状态：padding(准备状态：初始化成功、开始执行异步的任务)、
fullfilled(成功状态)、rejected(失败状态)==
Promise 本身是同步编程的，他可以管理异步操作的（重点），new Promise 的时候，会把传递的
函数立即执行
Promise 函数天生有两个参数，resolve(当异步操作执行成功，执行 resolve 方法),rejected(当异步
操作失败，执行 reject 方法)
then()方法中有两个函数，第一个传递的函数是 resolve,第二个传递的函数是 reject
ajax 中 false 代表同步，true 代表异步，如果使用异步，不等 ajax 彻底完成

14.数组方法
...

15.Vue 的组件之间是如何传递的？
父传子通过 props
子传父通过事件，父组件监听事件 (this.$emit(事件名, 传递给父组件的数据))，子组件触发事件；
非父子可以使用事件总线，也可以使用 Vuex

16.谈谈对 VUE 生命周期的理解
beforeCreate 阶段：vue 实例的挂载元素 el 和数据对象 data 都是 undefined，还没有初始化。
created 阶段：vue 实例的数据对象 data 有了，可以访问里面的

数据和方法，未挂载到 DOM，el
还没有
beforeMount 阶段：vue 实例的 el 和 data 都初始化了，但是挂载之前为虚拟的 dom 节点
mounted 阶段：vue 实例挂载到真实 DOM 上，就可以通过 DOM 获取 DOM 节点
beforeUpdate 阶段：响应式数据更新时调用，发生在虚拟 DOM 打补丁之前，适合在更新之前访问
现有的 DOM，比如手动移除已添加的事件监听器
updated 阶段：虚拟 DOM 重新渲染和打补丁之后调用，组成新的 DOM 已经更新，避免在这个钩
子函数中操作数据，防止死循环
beforeDestroy 阶段：实例销毁前调用，实例还可以用，this 能获取到实例，常用于销毁定时器，
解绑事件
destroyed 阶段：实例销毁后调用，调用后所有事件监听器会被移除，所有的子实例都会被销毁

17.请写出 vue 路由的钩子函数
全局钩子
beforeEach
beforeEach 函数有三个参数：
to: router 将要进入的路由对象
from: 当前导航将要离开的路由对象
next: 控制权函数
这类钩子主要作用于全局,一般用来判断权限,以及以及页面丢失时候需要执行的操作
路由独享钩子
beforeEnter
beforeLeave
主要用于写某个指定路由跳转时需要执行的逻辑
组件内钩子
beforeRouteEnter 在渲染该组件的对应路由前调用，该钩子中不

能访问 this，因为实例还未创建
beforeRouteUpdate 在当前路由改变，但是该组件被复用时调用，适用于动态路由的参数发生变化时
beforeRouteLeave 导航离开该组件的对应路由时调用，即要从当前页面去往其他页面时调用，可以用来做保存
拦截，例如提示用户信息尚未保存等

18.实现盒子水平垂直居中的实现方式
1.flex 布局，justify-content：center，align-items：center
2.绝对定位然后上右下左外边距都置 0，然后 margin：auto
3.绝对定位，left：50%，top：50%，margin-left：-元素宽度的一半，margin-top：-元素高度的一半（或者使用 transform:translate(-50%,-50%)来移动）

19.介绍一下 watch 监听的底层原理
监听的数据改变的时，watch 如何工作
watch 在一开始初始化的时候，会 读取 一遍 监听的数据的值，于是，此时 那个数据就收集到 watch 的 watcher 了
然后你给 watch 设置的 handler ，watch 会放入 watcher 的更新函数中
当数据改变时，通知 watch 的 watcher 进行更新，于是 你设置的 handler 就被调用

20.项目哪里用了 Vuex

21.项目哪里用了发布订阅
话题模块，关注楼主，取消关注

22.封装过哪些东西，方便说说原理嘛

23.element 二次封装

24.框架是怎么搭架

老大搭建的，我只是写逻辑

25.webpack 怎么配置的
老大搭建的，我只是写逻辑

26.http 和 https 区别
http 和 https

https：是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版本，通过 SSL 加密

http：超文本传输协议。是一个客服端和服务器端请求和应答的标准（tcp）,使浏览器更加高效，使网络传输减少

27.事件循环机制
event loop 过程
js 执行的时候会判断当前代码是同步还是异步，同步就进入同步消息队列中立即执行，异步就推进异步消息队列中等待执行
浏览器是多线程的，异步消息队列中的异步任务会被分配线程去执行
当所有同步任务执行完毕后，主线程空闲下来就会去到异步消息队列中看是否有异步任务达到可执行状态然后就将其推到同步消息队列中去执行(先微任务再宏任务以及队列先进先出机制)
上面三步操作循环就形成了事件循环（event loop）

28.Let 和 const 的区别

29.浏览器缓存如何实现的？
简单来说,浏览器缓存其实就是浏览器保存通过 HTTP 获取的所有资源,是浏览器将网络资源存储在本地的一种行为。

30.权限问题
首先说明每个用户登录的时候由于权限不同，所以展示的左侧菜单（菜单的每一列就会代表一个路由）是不一样的
1、当用户打开项目的时候，会请求 init 接口，这个接口是用户的

个人信息，其中包括一个权限的字段(permissionList)，里面是当前用户所对应的权限列表路由名称
2、前端渲染的菜单列表中每一个菜单都有一个 key 值和 permissionList 里的路由名称一样,所以当菜单列表渲染的时候会判断每一个菜单的 key 在 permissionList 里有没有，如果没有就不展示这个菜单。那这就是菜单列表的权限控制

31.登录注册
1、用户向服务器发送用户名和密码。
2、服务端收到请求，去验证用户名与密码
3、验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端。
4、客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage 里
5、用户随后的每一次请求，都会通过 Cookie，将 token 传回服务器。
6、服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据