一、CSS

1.说一下CSS的盒模型。

在HTML页面中的所有元素都可以看成是一个盒子

盒子的组成：内容content、内边距padding、边框border、外边距margin

盒模型的类型：

标准盒模型

margin + border + padding + content

IE盒模型

margin + content(border + padding)

控制盒模型的模式：box-sizing:content-box（默认值，标准盒模型）、border-box（IE盒模型）；

2.CSS选择器的优先级?

CSS的特性：继承性、层叠性、优先级

优先级：写CSS样式的时候，会给同一个元素添加多个样式，此时谁的权重高就显示谁的样式

标签、类/伪类/属性、全局选择器、行内样式、id、!important

!important > 行内样式 > id > 类/伪类/属性 > 标签 > 全局选择器

3.隐藏元素的方法有哪些?

display:none;

元素在页面上消失，不占据空间

opacity:0;

设置了元素的透明度为0，元素不可见，占据空间位置

visibility:hidden;

让元素消失，占据空间位置，一种不可见的状态

position:absolute;

clip-path

4.px和rem的区别是什么?

px是像素，显示器上给我们呈现画面的像素，每个像素的大小是一样，绝对单位长度

rem，相对单位，相对于html根节点的font-size的值，直接给html节点的font-size:62.5%;

1rem = 10px; （16px*62.5%=10px）

5.重绘重排有什么区别?

重排（回流）：布局引擎会根据所有的样式计算出盒模型在页面上的位置和大小

重绘：计算好盒模型的位置、大小和其他一些属性之后，浏览器就会根据每个盒模型的特性进行绘制

浏览器的渲染机制

对DOM的大小、位置进行修改后，浏览器需要重新计算元素的这些几何属性，就叫重排

对DOM的样式进行修改，比如color和background-color，浏览器不需要重新计算几何属性的时候，直接绘制了该元素的新样式，那么这里就只触发了重绘

6.让一个元素水平垂直居中的方式有哪些?

1.定位+margin

2.定位+transform

3.flex布局

4.grid布局

5.table布局

7.CSS的哪些属性哪些可以继承？哪些不可以继承?

CSS的三大特性：继承、层叠、优先级

子元素可以继承父类元素的样式

1.字体的一些属性：font

2.文本的一些属性：line-height

3.元素的可见性：visibility:hidden

4.表格布局的属性：border-spacing

5.列表的属性：list-style

6.页面样式属性：page

7.声音的样式属性

8.有没有用过预处理器?

预处理语言增加了变量、函数、混入等强大的功能

SASS LESS

二、JavaSscipt

1.JS由哪三部分组成?

ECMAScript：JS的核心内容，描述了语言的基础语法，比如var,for，数据类型（数组、字符串），

文档对象模型（DOM）：DOM把整个HTML页面规划为元素构成的文档
浏览器对象模型（BOM）：对浏览器窗口进行访问和操作
2.JS有哪些内置对象?
String Boolean Number Array Object Function Math Date RegExp...
Math
abs() sqrt() max() min()
Data
new Data() getYear()
Array
String
concat() length slice() split()
3.操作数组的方法有哪些?
push() pop() sort() splice() unshift() shift() reverse() concat() join() map() filter()
ervery() some() reduce() isArray() findIndex()
哪些方法会改变原数组?
push() pop() unshift() shift() sort() reverse() splice()
4.JS对数据类的检测方式有哪些?
typeof()
instanceof()
constructor
Object.prototype.toString.call()

 1686473056860

5.说一下闭包，闭包有什么特点?
什么是闭包？函数嵌套函数，内部函数被外部函数返回并保存下来时，就会产生闭包
特点：可以重复利用变量，并且这个变量不会污染全局的一种机制；这个变量是一直保存再内存中，不会被垃圾回收机制回收
缺点：闭包较多的时候，会消耗内存，导致页面的性能下降，在IE浏览器中才会导致内存泄漏
使用场景：防抖，节流，函数嵌套函数避免全局污染的时候
6.前端的内存泄漏怎么理解?
JS里已经分配内存地址的对象，但是由于长时间没有释放或者没办法清除，造成长期占用内存的现象，会让内存资源大幅浪费，最终导致运行速度慢，甚至崩溃的情况。
垃圾回收机制
因素：一些为生命直接赋值的变量；一些未清空的定时器；过度的闭包；一些引用元素没有被清除。
7.事件委托是什么?
又叫事件代理，原理就是利用了事件冒泡的机制来实现，也就是说把子元素的事件绑定到了父元素的身上
如果子元素组织了事件冒泡，那么委托也就不成立
组织事件冒泡：event.stopPropagation()
addEventListener('click',函数名，true/false) 默认是false（事件冒泡），true（事件捕获）
好处：提高性能，减少事件的绑定，也就减少了内存的占用。
8.基本数据类型和引用数据类型的区别?
基本数据类型：String Number Boolean undefined null
基本数据类型保存在栈内存当中，保存的就是一个具体的值
引用数据类型（复杂数据类型）：Object Function Array
保存在堆内存当中，声明一个引用类型的变量，它保存的是引用类型数据的地址
假如声明两个引用类型同时指向了一个地址的时候，修改其中一个那么另外一个也会改变
9.说一下原型链。
原型就是一个普通对象，它是为构造函数的实例共享属性和方法；所有实例中引用的原型都是同一个对象
使用prototype可以把方法挂在原型上，内存值保存一份
__proto__可以理解为指针，实例对象中的属性，指向了构造函数的原型（prototype）
10.new操作符具体做了什么?
1.先创建一个空对象
2.把空对象和构造函数通过原型链进行链接
3.把构造函数的this绑定到新的空对象身上
4.根据构建函数返回的类型判断，如果是值类型，则返回对象，如果是引用类型，就要返回这个引用类型
11.JS是如何实现继承的?
1.原型链继承
2.借用构造函数继承

3.组合式继承
4.ES6的class类继承
12.JS的设计原理是什么?
JS引擎 运行上下文 调用栈 事件循环 回调
13.JS中关于this指向的问题
1. 全局对象中的this指向
指向的是window
2. 全局作用域或者普通函数中的this
指向全局window
3. this永远指向最后调用它的那个对象
在不是箭头函数的情况下
4. new 关键词改变了this的指向
5. apply,call,bind
可以改变this指向，不是箭头函数
6. 箭头函数中的this
它的指向在定义的时候就已经确定了
箭头函数它没有this,看外层是否有函数，有就是外层函数的this，没有就是window
7. 匿名函数中的this
永远指向了window,匿名函数的执行环境具有全局性，因此this指向window
14.script标签里的async和defer有什么区别?
当没有async和defer这两个属性的时候，
浏览器会立刻加载并执行指定的脚本
有async
加载和渲染后面元素的过程将和script的加载和执行并行进行（异步）
有defer
加载和渲染后面元素的过程将和script的加载并行进行（异步），但是它的执行事件要等
所有元素解析完成之后才会执行
15.setTimeout最小执行时间是多少?
HTML5规定的内容：
setTimeout最小执行时间是4ms
setInterval最小执行时间是10ms
16.ES6和ES5有什么区别?
JS的组成：ECMAScript BOM DOM
ES5:ECMAScript5,2009年ECMAScript的第五次修订，ECMAScript2009
ES6:ECMAScript6,2015年ECMAScript的第六次修订，ECMAScript2015，是JS的下一个版本标准
17.ES6的新特性有哪些?
1.新增块级作用域（let,const）
不存在变量提升
存在暂时性死区的问题
块级作用域的内容
不能在同一个作用域内重复声明
2.新增了定义类的语法糖（class）
3.新增了一种基本数据类型（symbol）
4.新增了解构赋值
从数组或者对象中取值，然后给变量赋值
5.新增了函数参数的默认值
6.给数组新增了API
7.对象和数组新增了扩展运算符
8.Promise
解决回调地狱的问题。
自身有all,reject,resolve,race方法
原型上有then,catch
把异步操作队列化
三种状态：pending初始状态,fulfilled操作成功,rejected操作失败
状态：pending -> fulfilled;pending -> rejected 一旦发生，状态就会凝固，不会再变
async await
同步代码做异步的操作，两者必须搭配使用

async表明函数内有异步操作，调用函数会返回promise
await是组成async的表达式，结果是取决于它等待的内容，如果是promise那就是promise的结果，如果是普通函数就进行链式调用
await后的promise如果是reject状态，那么整个async函数都会中断，后面的代码不执行

9.新增了模块化（import,export）
10.新增了set和map数据结构
set就是不重复
map的key的类型不受限制
11.新增了generator
12.新增了箭头函数
不能作为构造函数使用，不能用new
箭头函数就没有原型
箭头函数没有arguments
箭头函数不能用call,apply,bind去改变this的执行
this指向外层第一个函数的this
18.call,aply,bind三者有什么区别?
都是改变this指向和函数的调用，call和apply的功能类似，只是传参的方法不同
call方法传的是一个参数列表
apply传递的是一个数组
bind传参后不会立刻执行，会返回一个改变了this指向的函数，这个函数还是可以传参的，bind()()
call方法的性能要比apply好一些，所以call用的更多一点
19.用递归的时候有没有遇到什么问题?
如果一个函数内可以调用函数本身，那么这个就是递归函数
函数内部调用自己
特别注意：写递归必须要有退出条件return
20.如何实现一个深拷贝?
深拷贝就是完全拷贝一份新的对象，会在堆内存中开辟新的空间，拷贝的对象被修改后，原对象不受影响
主要针对的是引用数据类型
1.扩展运算符
2.JSON.parse(JSON.stringify())
3.利用递归函数实现
21.说一下事件循环。
JS是一个单线程的脚本语言
主线程 执行栈 任务队列 宏任务 微任务
主线程先执行同步任务，然后才去执行任务队列里的任务，如果在执行宏任务之前有微任务，那么要先执行微任务
全部执行完之后等待主线程的调用，调用完之后再去任务队列中查看是否有异步任务，这样一个循环往复的过程就是事件循环!
22.ajax是什么？怎么实现的?
创建交互式网页应用的网页开发技术
在不重新加载整个网页的前提下，与服务器交换数据并更新部分内容
通过XmlHttpRequest对象向服务器发送异步请求，然后从服务器拿到数据，最后通过JS操作DOM更新页面
1.创建XmlHttpRequest对象 xmh
2.通过xmh对象里的open()方法和服务器建立连接
3.构建请求所需的数据，并通过xmh对象的send()发送给服务器
4.通过xmh对象的onreadystate chansge事件监听服务器和你的通信状态
5.接收并处理服务器响应的数据结果
6.把处理的数据更新到HTML页面上
23.get和post有什么区别?
1.get一般是获取数据，post一般是提交数据
2.get参数会放在url上，所以安全性比较差，post是放在body中
3.get请求刷新服务器或退回是没有影响的，post请求退回时会重新提交数据
4.get请求时会被缓存,post请求不会被缓存
5.get请求会被保存在浏览器历史记录中,post不会
6.get请求只能进行url编码，post请求支持很多种
24.promise的内部原理是什么？它的优缺点是什么?
Promise对象，封装了一个异步操作并且还可以获取成功或失败的结果
Promise主要就是解决回调地狱的问题，之前如果异步任务比较多，同时他们之间有相互依赖的关系，
就只能使用回调函数处理，这样就容易形成回调地狱，代码的可读性差，可维护性也很差
有三种状态：pending初始状态 fulfilled成功状态 rejected失败状态
状态改变只会有两种情况，
pending -> fulfilled; pending -> rejected 一旦发生，状态就会凝固，不会再变
首先就是我们无法取消promise，一旦创建它就会立即执行，不能中途取消
如果不设置回调，promise内部抛出的测u哦鸣就无法反馈到外面
若当前处于pending状态时，无法得知目前在哪个阶段。
原理：
构造一个Promise实例，实例需要传递函数的参数，这个函数有两个形参，分别都是函数类型，一个是resolve一个是reject
promise上还有then方法，这个方法就是来指定状态改变时的确定操作，resolve是执行第一个函数，reject是执行第二个函数
25.promise和async await的区别是什么?
1.都是处理异步请求的方式
2.promise是ES6，async await 是ES7的语法
3.async await是基于promise实现的，他和promise都是非阻塞性的
优缺点：
1.promise是返回对象我们要用then，catch方法去处理和捕获异常，并且书写方式是链式，容易造成代码重叠，不好维护，async await 是通过tra catch进行捕获异常
2.async await最大的优点就是能让代码看起来像同步一样，只要遇到await就会立刻返回结果，然后再执行后面的操作
promise.then()的方式返回，会出现请求还没返回，就执行了后面的操作
26.浏览器的存储方式有哪些?
1.cookies
H5标准前的本地存储方式
兼容性好，请求头自带cookie
存储量小，资源浪费，使用麻烦（封装）
2.localstorage

H5加入的以键值对为标准的方式
操作方便，永久存储，兼容性较好
保存值的类型被限定，浏览器在隐私模式下不可读取，不能被爬虫

3.sessionstorage
当前页面关闭后就会立刻清理，会话级别的存储方式

4.indexedDB
H5标准的存储方式，，他是以键值对进行存储，可以快速读取，适合WEB场景

27.token存在sessionstorage还是loaclstorage?
token：验证身份的令牌，一般就是用户通过账号密码登录后，服务端把这些凭证通过加密等一系列操作后得到的字符串
1.存loaclstorage里，后期每次请求接口都需要把它当作一个字段传给后台
2.存cookie中，会自动发送，缺点就是不能跨域
如果存在localstorage中，容易被XSS攻击，但是如果做好了对应的措施，那么是利大于弊
如果存在cookie中会有CSRF攻击

28.token的登录流程。
1.客户端用账号密码请求登录
2.服务端收到请求后，需要去验证账号密码
3.验证成功之后，服务端会签发一个token，把这个token发送给客户端
4.客户端收到token后保存起来，可以放在cookie也可以是localstorage
5.客户端每次向服务端发送请求资源的时候，都需要携带这个token
6.服务端收到请求，接着去验证客户端里的token，验证成功才会返回客户端请求的数据

29.页面渲染的过程是怎样的?
DNS解析
建立TCP连接
发送HTTP请求
服务器处理请求
渲染页面
浏览器会获取HTML和CSS的资源，然后把HTML解析成DOM树
再把CSS解析成CSSOM
把DOM和CSSOM合并为渲染树
布局
把渲染树的每个节点渲染到屏幕上（绘制）
断开TCP连接

30.DOM树和渲染树有什么区别?
DOM树是和HTML标签一一对应的，包括head和隐藏元素
渲染树是不包含head和隐藏元素

31.精灵图和base64的区别是什么?
精灵图：把多张小图整合到一张大图上，利用定位的一些属性把小图显示在页面上，当访问页面可以减少请求，提高加载速度
base64：传输8Bit字节代码的编码方式，把原本二进制形式转为64个字符的单位，最后组成字符串
base64是会和html css一起下载到浏览器中，减少请求，减少跨域问题，但是一些低版本不支持，若base64体积比原图片大，不利于css的加载。

32.svg格式了解多少?
基于XML语法格式的图像格式，可缩放矢量图，其他图像是基于像素的，SVG是属于对图像形状的描述，本质是文本文件，体积小，并且不管放大多少倍都不会失真
1.SVG可直接插入页面中，成为DOM一部分，然后用JS或CSS进行操作
```
<svg></svg>
```
2.SVG可作为文件被引入
```
<img src="pic.svg" />
```
3.SVG可以转为base64引入页面

33.了解过JWT吗?
JSON Web Token 通过JSON形式作为在web应用中的令牌，可以在各方之间安全的把信息作为JSON对象传输
信息传输、授权
JWT的认证流程
1.前端把账号密码发送给后端的接口
2.后端核对账号密码成功后，把用户id等其他信息作为JWT 负载，把它和头部分别进行base64编码拼接后签名，形成一个JWT（token）。
3.前端每日请求时都会把JWT放在HTTP请求头的Authorization字段内
4.后端检查是否存在，如果存在就验证JWT的有效性（签名是否正确，token是否过期）
5.验证通过后后端使用JWT中包含的用户信息进行其他的操作，并返回对应结果
简洁、包含性、因为Token是JSON加密的形式保存在客户端，所以JWT是跨语言的，原则上是任何web形式都支持。

34.npm的底层环境是什么?
node package manager,node的包管理和分发工具，已经成为分发node模块的标准，是JS的运行环境
npm的组成：网站、注册表、命令行工具

35.HTTP协议规定的协议头和请求头有什么?
1.请求头信息：
Accept:浏览器告诉服务器所支持的数据类型
Host:浏览器告诉服务器我想访问服务器的哪台主机
Referer:浏览器告诉服务器我是从哪里来的（防盗链）
User-Agent:浏览器类型、版本信息
Date:浏览器告诉服务器我是什么时候访问的
Connection:连接方式
Cookie
X-Request-With:请求方式
2.响应头信息：
Location:这个就是告诉浏览器你要去找谁
Server:告诉浏览器服务器的类型
Content-Type:告诉浏览器返回的数据类型
Refresh:控制了的定时刷新

36.说一下浏览器的缓存策略。
强缓存（本地缓存）、协商缓存（弱缓存）
强缓：不发起请求，直接使用缓存里的内容，浏览器把JS，CSS，image等存到内存中，下次用户访问直接从内存中取，提高性能

协缓：需要像后台发请求，通过判断来决定是否使用协商缓存，如果请求内容没有变化，则返回304，浏览器就用缓存里的内容
强缓存的触发：
HTTP1.0:时间戳响应标头
HTTP1.1:Cache-Control响应标头
协商缓存触发：
HTTP1.0:请求头：if-modified-since 响应头：last-modified
HTTP1.1:请求头：if-none-match 响应头：Etag

37.说一下什么是“同源策略”?
http:// www. aaa.com:8080/index/vue.js
协议 子域名 主域名 端口号 资源
同源策略是浏览器的核心，如果没有这个策略就会遭受网络攻击
主要指的就是协议+域名+端口号三者一致，若其中一个不一样则不是同源，会产生跨域
三个允许跨域加载资源的标签：img link script
跨域是可以发送请求，后端也会正常返回结果，只不过这个结果被浏览器拦截了!
JSONP
CORS
websocket
反向代理

38.防抖和节流是什么?
都是应对页面中频繁触发事件的优化方案
防抖:避免事件重复触发
使用场景:1.频繁和服务端交互 2.输入框的自动保存事件
节流:把频繁触发的事件减少,每隔一段时间执行
使用场景:scroll事件

39.解释一下什么是json?
JSON是一种纯字符串形式的数据，它本身不提供任何方法，适合在网络中进行传输
JSON数据存储在.json文件中，也可以把JSON数据以字符串的形式保存在数据库、Cookise中
JS提供了JSON.parse() JSON.stringify()
什么时候使用json：定义接口；序列化；生成token；配置文件package.json

40.当数据没有请求过来的时候，该怎么做?
可以在渲染数据的地方给一些默认的值
if判断语句

41.有没有做过无感登录?
1.在相应其中拦截，判断token返回过期后，调用刷新token的接口
2.后端返回过期时间，前端判断token的过期时间，去调用刷新token的接口
3.写定时器，定时刷新token接口
流程：
1.登录成功后保存token 和 refresh_token
2.在响应拦截器中对401状态码引入刷新token的api方法调用
3.替换保存本地新的token
4.把错误对象里的token替换
5.再次发送未完成的请求
6.如果refresh_token过期了，判断是否过期，过期了就清楚所有token重新登录

42.大文件上传是怎么做的?
分片上传：
1.把需要上传的文件按照一定的规则，分割成相同大小的数据块
2.初始化一个分片上传任务，返回本次分片上传的唯一标识
3.按照一定的规则把各个数据块上传
4.发送完成后，服务端会判断数据上传的完整性，如果完整，那么就会把数据库合并成原始文件
断点续传：
服务端返回，从哪里开始 浏览器自己处理

三、HTML5CSS3

1.语义化的理解。

在写HTML页面结构时所用的标签有意义

头部用head 主体用main 底部用foot...

怎么判断页面是否语义化了?

把CSS去掉，如果能够清晰的看出来页面结构，显示内容较为正常

为什么要选择语义化?

1.让HTML结构更加清晰明了

2.方便团队协作，利于开发

3.有利于爬虫和SEO

4.能够让浏览器更好的去解析代码

5.给用户带来良好的体验

2.H5C3有哪些新特性?

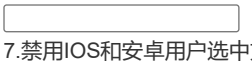
H5的新特性：

1.语义化的标签

2.新增音频视频

3.画布canvas

4.数据存储localstorage sessionstorage

5.增加了表单控件 email url search...
6.拖拽释放API
CSS3的新特性：
1.新增选择器：属性选择器、伪类选择器、伪元素选择器
2.增加了媒体查询
3.文字阴影
4.边框
5.盒子模型box-sizing
6.渐变
7.过度
8.自定义动画
9.背景的属性
10.2D和3D
3.rem是如何做适配的?
rem是相对长度，相对于根元素（html）的font-size属性来计算大小，通常来做移动端的适配
rem是根据根元素font-size计算值的倍数
比如html上的font-size:16px，给div设置宽为1.5rem,1.2rem = 16px*1.2 = 19.2px.
4.解决了哪些移动端的兼容问题?
1.当设置样式overflow:scroll/auto时，IOS上的华东会卡顿
-webkit-overflow-scrolling:touch;
2.在安卓环境下placeholder文字设置行高时会偏上
input有placeholder属性的时候不要设置行高
3.移动端字体小于12px时异常显示
应该先把在整体放大一倍，然后再用transform进行缩小
4.ios下input按钮设置了disabled属性为true显示异常
input[typy=button]{
opcity:1
}
5.安卓手机下取消语音输入按钮
input::-webkit-input-speech-button{
display:none
}
6.IOS下取消input输入框在输入引文首字母默认大写

7.禁用IOS和安卓用户选中文字
添加全局CSS样式：-webkit-user-select:none
8.禁止IOS弹出各种窗口
-webkit-touch-callout:none
9.禁止IOS识别长串数字为电话
添加meta属性

四、Vue
1.v-if和v-show的区别?
都可以控制元素的显示和隐藏
1.v-show时控制元素的display值来让元素显示和隐藏；v-if显示隐藏时把DOM元素整个添加和删除
2.v-if有一个局部编译/卸载的过程，切换这个过程中会适当的销毁和重建内部的事件监听和子组件；v-show只是简单的css切换
3.v-if才是真正的条件渲染；v-show从false变成true的时候不会触发组件的声明周期，v-if会触发声明周期
4.v-if的切换效率比较低 v-show的效率比较高
2.如何理解MVVM的?
是Model-View-ViewModel的缩写。前端开发的架构模式
M：模型，对应的就是data的数据
V：视图，用户界面，DOM
VM：视图模型：Vue的实例对象，连接View和Model的桥梁
核心是提供对View和ViewModel的双向数据绑定，当数据改变的时候，ViewModel能监听到数据的变化，自动更新视图，当用户操作视图的时候，ViewModel也可以监听到视图的变化，然后通知数据进行改动，这就实现了双向数据绑定
ViewModel通过双向绑定把View和Model连接起来，他们之间的同步是自动的，不需要认为干涉，所以我们只需要关注业务逻辑即可，不需要操作DOM，同时也不需要关注数据的状态问题，因为她是由MVVM统一管理

3.v-for中的key值的作用是什么?
key属性是DOM元素的唯一标识
作用:
1.提高虚拟DOM的更新
2.若不设置key，可能会触发一些bug
3.为了触发过度效果
4.说一下你对vue生命周期的理解。
组件从创建到销毁的过程就是它的生命周期
创建
beforeCreat
在这个阶段属性和方法都不能使用
created
这里时实例创建完成之后，在这里完成了数据监测，可以使用数据，修改数据，不会触发updated，也不会更新视图
挂载
beforeMount
完成了模板的编译，虚拟DOM也完成创建，即将渲染，修改数据，不会触发updated
Mounted
把编译好的模板挂载到页面，这里可以发送异步请求也可以访问DOM节点
更新
beforeUpdate
组件数据更新之前使用，数据是新的，页面上的数据时旧的，组件即将更新，准备渲染，可以改数据
updated
render重新做了渲染，这时数据和页面都是新的，避免在此更新数据
销毁
beforeDestroy
实例销毁前，在这里实例还可以用，可以清楚定时器等等
destroyed
组件已经被销毁了，全部都销毁
使用了keep-alive时多出两个周期:
activited
组件激活时
deactivited
组件被销毁时
5.在created和mounted去请求数据，有什么区别?
created：在渲染前调用，通常先初始化属性，然后做渲染
mounted：在模板渲染完成后，一般都是初始化页面后，在对元素节点进行操作
在这里请求数据可能会出现闪屏的问题，created里不会
一般用created比较多
请求的数据对DOM有影响，那么使用created
如果请求的数据对DOM无关，可以放在mounted
6.vue中的修饰符有哪些?
1.事件修饰符
.stop 组织冒泡
.prevent 组织默认行为
.capture 内部元素触发的事件先在次处理
.self 只有在event.target是当前元素时触发
.once 事件只会触发一次
.passive 立即触发默认行为
.native 把当前元素作为原生标签看待
2.按键修饰符
.keyup 键盘抬起
.keydown 键盘按下
3.系统修饰符
.ctrl
.alt
.meta
4.鼠标修饰符

.left 鼠标左键
.right 鼠标右键
.middle 鼠标中键
5.表单修饰符
.lazy 等输入完之后再显示
.trim 删除内容前后的空格
.number 输入是数字或转为数字
7.elementui是怎么做表单验证的?
1.在表单中加rules属性，然后再data里写校验规则
2.内部添加规则
3.自定义函数校验
8.vue如何进行组件通信?
1.父传子
props
父组件使用自定义属性，然后子组件使用props
refs对象上
2.子传父

router.push({name:'index',params:{id:item.id}})
this. router.push({name:'/index/ParseError: KaTeX parse error: Expected 'EOF', got '}' at position 11: {item.id}'}) 路…
router.push({
name:'index',
query:{id:item.id}
})
12.vue路由的hash模式和history模式有什么区别?
1.hash的路由地址上有#号，history模式没有
2.在做回车刷新的时候，hash模式会加载对应页面，history会报错404
3.hash模式支持低版本浏览器，history不支持，因为是H5新增的API
4.hash不会重新加载页面，单页面应用必备
5.history有历史记录，H5新增了pushState和replaceState()去修改历史记录，并不会立刻发送请求
6.history需要后台配置
13.路由拦截是怎么实现的?
路由拦截 axios拦截
需要在路由配置中添加一个字段，它是用于判断路由是否需要拦截
{
name:'index',
path:'/index',
component:Index,
meta:{
requirtAuth:true
}
}
router.beforeEach((to,from,next) => {
if(to.meta.requirtAuth){
if(store.satte.token){
next()
}else{

```
            }
        }
    })
```

14.说一下vue的动态路由。

要在路由配置里设置meat属性，扩展权限相关的字段，在路由导航守卫里通过判断这个权限标识，实现路由的动态增加和跳转

根据用户登录的账号，返回用户角色

前端再根据角色，跟路由表的meta.role进行匹配

把匹配搭配的路由形成可访问的路由

15.如何解决刷新后二次加载路由?

1.window.location.reload()

2.matcher

```
const router = createRouter()
export function resetRouter(){
    const newRouter = creatRouter()
    router.matcher = newRouter.matcher
}
```

16.vuex刷新数据会丢失吗？怎么解决?

vuex肯定会重新获取数据，页面也会丢失数据

1.把数据直接保存在浏览器缓存里（cookie localstorage sessionstorage）

2.页面刷新的时候，再次请求数据，达到可以动态更新的方法

监听浏览器的刷新书简，在刷新前把数据保存到sessionstorage里，刷新后请求数据，请求到了用vuex，如果没有那就用sessionstorage里的数据

17.computed和watch的区别?

1.computed是计算属性，watch是监听，监听的是data中数据的变化

2.computed是支持缓存，依赖的属性值发生变化，计算属性才会重新计算，否则用缓存；watch不支持缓存

3.computed不支持异步，watch是可以异步操作

4.computed是第一次加载就监听，watch是不监听

5.computed函数中必须有return watch不用

18.vuex在什么场景会去使用？属性有哪些?

state	存储变量
getters	state的计算属性
mutations	提交更新数据的方法
actions	和mutations差不多，他是提交mutations来修改数据，可以包括异步操作
modules	模块化vuex

使用场景：

用户的个人信息、购物车模块、订单模块

19.vue的双向数据绑定原理是什么?

通过数据劫持和发布订阅者模式来实现，同时利用Object.defineProperty()劫持各个属性的setter和getter，

在数据发生改变的时候发布消息给订阅者，触发对应的监听回调渲染视图，也就是说数据和视图时同步的，数据发生改变，视图跟着发生改变，视图改变，数据也会发生改变。

第一步：需要observer的数据对象进行递归遍历，包括子属性对象的属性，都加上setter和getter

第二步：compile模板解析指令，把模板中的变量替换成数据，然后初始化渲染视图，同时把每个指令对应的节点绑定上更新函数，添加订阅者，如果数据变化，收到通知，更新

第三步：Watcher订阅者是Observer和Compile之间的通信桥梁，作用：

1.在自身实例化的时候忘订阅器内添加自己

2.自身要有一个update()方法

3.等待属性变动时，调用自身的update方法，触发compile这种的回调

第四步：MVVM作为数据绑定的入口，整合了observer、compile和watcher三者，通过observer来监听自己的数据变化，通过compile解析模板指令，最后利用watcher把obse

20.了解diff算法和虚拟DOM吗?

虚拟DOM，描述元素和元素之间的关系，创建一个JS对象

如果组件内有响应的数据，数据发生改变的时候，render函数会生成一个新的虚拟DOM，这个新的虚拟DOM会和旧的虚拟DOM进行比对，找到需要修改的虚拟DOM内容，然后去对应

diff算法就是虚拟DOM的比对时用的，返回一个patch对象，这个对象的作用就是存储两个节点不同的地方，最后用patch里记录的信息进行更新真实DOM

步骤：

1.JS对象表示真实的DOM结构，要生成一个虚拟DOM，再用虚拟DOM构建一个真实DOM树，渲染到页面

2.状态改变生成新的虚拟DOM，跟就得虚拟DOM进行比对，这个比对的过程就是DIFF算法，利用patch记录差异

3.把记录的差异用在第一个虚拟DOM生成的真实DOM上，视图就更新了。

21.vue和jquery的区别是什么?

1.原理不同

vue就是数据绑定；jq是先获取dom再处理

2.着重点不同

vue是数据驱动，jq是着重于页面

3.操作不同

4.未来发展不同

22.vuex的响应式处理。

vuex是vue的状态管理工具

vue中可以直接触发methods中的方法，vuex是不可以的。未来处理异步，当触发事件的时候，会通过dispatch来访问actions中的方法，actions中的commit会触发mutation

Vue.use(vuex)，调用install方法，通过applyMixin(vue)在任意组件内执行this.$store就可以访问到store对象。

vuex的state是响应式的，借助的就是vue的data，把state存到vue实例组件的data中

23.vue中遍历全局的方法有哪些?

1.普通遍历，对象.forEach()

```
arr.forEach(function(item,index,arr){
    console.log(item,index)
})
```

2.对元素统一操作 对象.map()

```
var newarr = arr.map(function(item){
    return item+1
})
```

3.查找符合条件的元素 对象.filter()

```
arr.filter(function(item){
```

```
            if(item > 2){
                return false
            }else{
                return true
            }
        })
```

4.查询符合条件的元素，返回索引 对象.findindex()

```
        arr.finindex(function(item){
            if(item>1){
                return true
            }else{
                return false
            }
        })
```

对象.evening() 遇到不符合的对象会停止

对象.some() 找到符合条件的元素就停止

24.如何搭建脚手架?

下载：node cnpm webpack vue-cli

创建项目：

1.找到对应的文件，然后利用node指令创建（cmd）

2.vue init webpack xxxx

3.回车项目描述

4.作者回车

5.选择vue build

6.回车

7.输入n

8.不按照yarn

9.输入npm run dev

25.如何封装一个组件?

1.使用Vue.extend()创建一个组件

2.使用Vue.components()方法注册组件

3.如果子组件需要数据，可以在props中接收定义

4.子组件修改好数据，要把数据传递给父组件，可以用emit()方法

原则：

把功能拆开

尽量让组件原子化，一个组件做一件事情

容器组件管数据，展示组件管视图

26.封装一个可复用的组件，需要满足什么条件?

1.低耦合，组件之间的依赖越小越好

2.最好从父级传入信息，不要在公共组件中请求数据

3.传入的数据要进行校验

4.处理事件的方法写在父组件中

27.vue的过滤器怎么使用?

vue的特性，用来对文本进行格式化处理

使用它的两个地方，一个是插值表达式，一个是v-bind

分类：

1.全局过滤器

```
            Vue.filter('add',function(v){
                return v < 10 ? '0' + v : v
            })
            <div>{{33 | add}}</div>
```

2.本地过滤器

```
            和methods同级
            filter:{
                add:function(v){
                    return v < 10 ? '0' + v : v
                }
            }
```

28.vue中如何做强制刷新?

1.localtion.reload()

2.this.$router.go(0)

3.provide和inject

29.vue3和vue2有哪些区别?

1.双向数据绑定的原理不同

2.是否支持碎片

3.API不同

4.定义数据变量方法不同

5.生命周期的不同

6.传值不同

7.指令和插槽不同

8.main.js不同

30.vue的性能优化怎么做?

1.编码优化

不要把所有数据都放在data中

v-for时给每个元素绑定事件用事件代理

keep-alive缓存组件

尽可能拆分组件，提高复用性、维护性

key值要保证唯一
合理使用路由懒加载，异步组件
数据持久化存储的使用尽量用防抖、节流优化
2.加载优化
按需加载
内容懒加载
图片懒加载
3.用户体验
骨架屏
4.SEO优化
预渲染
服务端渲染ssr
5.打包优化
CDN形式加载第三方模块
多线程打包
抽离公共文件
6.缓存和压缩
客户端缓存、服务端缓存
服务端Gzip压缩

31.首屏优化该如何去做?
1.使用路由懒加载
2.非首屏组件使用异步组件
3.首屏不中要的组件延迟加载
4.静态资源放在CDN上
5.减少首屏上JS、CSS等资源文件的大小
6.使用服务端渲染
7.简历减少DOM的数量和层级
8.使用精灵图请求
9.做一些loading
10.开启Gzip压缩
11.图片懒加载

32.vue3的性能为什么比vue2好?
1.diff算法的优化
2.静态提升
3.事件侦听缓存

33.vue3为什么使用proxy?
1.proxy可以代理整个对象，defineproperty只代理对象上的某个属性
2.proxy对代理对象的监听更加丰富
3.proxy代理对象会生成新的对象，不会修改被代理对象本身
4.proxy补兼容ie浏览器

34.说一下你对组件的理解。
可以重复使用的vue实例，独一无二的组件名称
可以抽离单独的公共模块
提高代码的复用率

35.你是如何规划项目文件的?
public
图标、index.html、img
src
api
assets
components
按分类再次划分子目录
plugins
router
static
styles
utils
views
App.vue
main.js
package.json
vue.config.js

36.是否使用过nuxt.js?
是基于vue的应用框架，关注的是渲染，可以开发服务端渲染应用的配置
SSR：服务端渲染
好处：
SSR生成的是有内容的HTML页面，有利于搜索引擎的搜索
优化了首屏加载时间
SEO：优化搜索引擎
SPA的应用不利于搜索引擎SEO的操作

37.SEO如何优化?
1.SSR
2.预渲染 prerender-spa-plugin

五、Echarts

1.echarts有用过吗？常用的组件有哪些?

title标题组件 show text link
toolbox工具栏 导出图片 数据视图 切换 缩放 show orient feature
tooltip tigger 触发类型
markPoint标注点
markLine图标的标线

六、Uni-APP
1.uni-app有没有做过分包?
优化小程序的下载和启动速度
小程序启动默认下载主包并启动页面，当用户进入分包时，才会下载对应的分包，下载完进行展示

七、Weabpack
1.webpack打包和不打包的区别?
1.运行效率
2.对基础的支持不够
2.webpack是怎么打包的，babel是做什么的?
webpack会把js css image看作一个模块，用import/require引入
找到入口文件，通过入口文件找到关联的依赖文件，把他们打包到一起
把bundle文件，拆分成多个小的文件，异步按需加载所需要的文件
如果一个被多个文件引用，打包时只会生成一个文件
如果引用的文件没有调用，不会打包，如果引入的变量和方法没有调用也不会打包
对于多个入口文件，加入引入了相同的代码，可以用插件把他抽离到公共文件中

八、Git
1.git如何合并、拉取代码?
拉取代码 git pull '仓库地址'
查看状态 git sattus
提交到本地缓存区 git add .
提交本地仓库 git commit -m '修改描述'
提交到远程仓库 git push '仓库地址' master
创建分支 git branch -b xxx
合并分支 git merge '合并分支的名字'
2.git如何解决冲突问题?
1.两个分支中修改了同一个文件
2.两个分支中修改了同一个文件的名字
1.解决：当前分支上，直接修改代码 add commit
2.解决：在本地当前分支上，修改冲突代码 add commit push

九、HR
1.你的离职原因是什么?
疫情 社保 薪资问题 个人发展 技术提升 家庭因素
2.工作到现在，项目中遇到最难的问题是什么？怎么解决的?
1.不要回答，没有问题
2.不要说一些常见的简单的问题，比如：数据请求不过来、渲染页面时出现了问题、跳转路由不会...
首先应该时自行去查找资料寻求解决办法，然后再去请教同时或者组长
3.你的优势在哪里?
1.尽量不要暴露自己的缺点
2.不要过度美化自己
4.如何协同工作?
1.开发前会开个会议，最后形成一个开发文档
2.利用工具保证项目的正常进度，规范化