1, Js数据类型有哪些？

数值、字符串、布尔、undefined、null、数组、对象、函数

2, 引用类型和值类型的区别

- 值类型存在于栈中, 存取速度快 引用类型存在于堆,存取速度慢

- 值类型复制的是值本身 引用类型复制的是指向对象的指针

- 值类型结构简单只包含基本数据, 引用类型结构复杂,可以实现多层嵌套

3, 你的项目需要在页面上播放视频，用什么播放？怎么防止用户下载视频？

可以使用video标签播放视频,

通过DOM事件禁止下载, 可以在页面的鼠标右击事件和F12键盘事件的回调中return false 以屏蔽用户操作

4, 如何防止你的页面数据被抓取(复制或保存) ?

(1, 阻止鼠标右击事件的默认行为(弹出菜单)

(2, 阻止F12按键的点按事件默认行为(检查元素)

(3, 组件ctrl+C组合按键的默认行为 (复制)

5, 怎样实现在动画结束之后执行一段代码？

过度动画：监听transitionend事件。

关键帧动画：监听animationend事件。

Js动画：自定义js动画时，直接在停止动画的代码之后添加要执行的代码，如果使用js动画工具库(velocity)，则把要执行的代码写在动画结束时的回调函数中。

6, Promise有哪些使用场景?

(1, 在页面打开时,要同时执行多个ajax请求,可以使用promise处理多异步任务并发执行

(2, 有些ajax请求之间存在依赖关系,需要顺序执行,造成结构嵌套,可以使用promise解决异步任务多层嵌套的问题, 实现链式调用

(3, 在项目中封装网络请求时,使用peomise封装ajax请求并返回peomise对象

7, 浏览器中事件传播的 流程是什么？怎样阻止事件的传播？怎样阻止事件的默认行为。

事件触发后，先从根元素到叶子元素逐层捕获，然后再从叶子元素到根元素逐层冒泡。 e.stopPropergation , e.preventDefault

8, Js函数中的this有哪些指向？怎样改变函数中this的指向。

一般情况下，通过谁调用，就指向谁。

在js全局作用域, this指向window

在对象中,this指向这个对象本身

构造函数中，this是正在创建的对象。

在事件函数中,this指向事件目标

(注意: 在计时器中this会被还原成window或置空,但箭头函数可以保留this指向)

可用通过call(), apply(), bind()改变this的指向

apply，和call一样,修改指向的同时调用函数，唯一的区别是，传参方式不同，aplly需要提供一个数组。

bind修改this指向时不会调用函数，而是生成一个新的函数，新的函数和原函数代码一样，但是里面的this是绑定过的。

9, 如何阻止form表单提交？

监听form表单submit事件，在事件处理函数内部return false 或者阻止浏览器默认行为

10, 数组有哪些常用处理方法，字符串、数字有哪些常用处理方法。

数组：join,splice,push,slice,indexof,sort,concat,reverse,map,reduce,some,every,filter,foreach。

字符串:

split,indexof,substr,substring,replace,trim,toUppercase,startsWith,

数字:tofixed。

Math: floor,ceil,round,random,abs,sqrt,pow,max,sin,

11, 什么是原型和原型链？

Js中的对象都有一个属性叫做\_\_ptoto\_\_(也是一个对象)，表示对象的原型。当访问对象中的属性或方法时，首先在对象本身中寻找，如果找不到则会在原型中寻找，原型中也找不到时会在原型的原型中寻找，直到最顶层为止。

js中的类(构造函数)都有一个prototype的属性，表示本类的原型类型，通过这个类实例化的对象(这个构造函数创建的对象)， \_\_proto\_\_都指向本类的prototype，从而实现了类方法的共享。一个类的prototype也是一个对象，它也有\_\_proto\_\_，把它的\_\_proto\_\_指向另一个类的prototype时，那么这个类的对象就能访问另一个类中的方法，从而实现了方法的继承。

A类的prototype.\_\_proto\_\_指向另一个类B，B的prototype.\_\_proto\_\_又可以指向C，这种结构叫做原型链。

12, 什么是JS严格模式？怎么进入干戈模式？严格模式下有哪些限制？

严格模式是JS的一种特殊执行模式，设立”严格模式”的目的，主要有以下几个：

1，消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

2， 消除代码运行的一些不安全之处，保证代码运行的安全；

3，提高编译器效率，增加运行速度；

4， 为未来新版本的Javascript做好铺垫。

在函数中或全局域的开头添加字符串”user strict”进入严格模式。

严格模式主要有以下限制：

变量必须声明后再使用

函数的参数不能有同名属性，否则报错

不能使用with语句

不能对只读属性赋值，否则报错

不能使用前缀0表示八进制数，否则报错

不能删除不可删除的属性，否则报错

不能删除变量delete prop，会报错，只能删除属性delete global[prop]

eval不会在它的外层作用域引入变量

eval和arguments不能被重新赋值

arguments不会自动反映函数参数的变化

不能使用arguments.callee

不能使用arguments.caller

禁止this指向全局对象

不能使用fn.caller和fn.arguments获取函数调用的堆栈

增加了保留字（比如protected、static和interface）

13, Js原生Ajax实现流程

(1)创建XMLHttpRequest请求对象

(2)open方法指定请求方式、请求路径、同步异步

(3)设置响应HTTP请求状态变化的函数

(4)send方法发送请求

(5)响应成功使用JavaScript和DOM实现局部刷新

14, 闭包是什么？有什么作用？

当一个函数A的作用域被内部的B函数引用时，A函数的作用域就会被B函数闭包，当A函数执行完毕时，A函数的作用域也不会释放。

闭包可以实现对象的私有属性和私有方法。

闭包可以封装变量，从简减少对全局作用域的污染。

15, 什么是函数防抖和函数节流？

函数防抖：对于频繁触发的事件，如果只希望其最后一次（或第一次）执行绑定函数的执行，则需要使用函数防抖。

函数节流：对于频繁触发的事件，希望其按照一定的频率进行绑定函数调用，则使用函数节流。

函数防抖和节流都可以通过settimeout实现。

16, 栈和队列的区别是什么?js中怎样实现栈结构？

栈和队列都是线型数据结构，栈只有一个入口，同时也是出口，所以数据遵循先进后出，后进先出的规则。队列一侧是入口，另一侧是出口，所以数据先进先出，后进后出。

Js中的数组可以实现栈和队列。Push和pop方法是一对栈操作，push和shift是一对队列操作。

17, 什么是深拷贝和浅拷贝？

浅拷贝就是只复制数组（对象）本身，而不复制其内容（引用类型的数据内容），最终两个数组中指向同一套数据。深拷贝则是既赋值本身也赋值内容。

Js中对于引用类型的数据，默认进行的都是浅拷贝。

18, ES6中新增加了哪些特性？

箭头函数，字符串模板，let块级作用域声明方式，const常量，class声明类，结构赋值，Promise，ES6模块化。

19, 箭头函数和普通函数的区别

- 箭头函数内部this跟函数所在上下文this保持一致

- 没有arguments参数集合，可以用rest替代

- 不能使用call、bind、apply来改变this的指向

20, 使用axios发起ajax请求无法携带cookie,什么原因? 如何解决?

axios默认跨域请求不使用凭证,当服务器在响应头中设置了cookie后, axios会默认隐藏这部分信息,

设置**axios.defaults.withCredentials = true；**即可

21, 什么是跨域请求，怎样实现跨域请求。

一个页面发送到非本身来源的请求叫做跨域请求，浏览器只禁止ajax跨域请求。

三种跨域请求的方式：

1，cors，服务器在响应头中添加access control allow origin字段，浏览器在收到请求之后就会认为本次请求时允许跨域的。

2，JSONP,浏览器使用创建script标签的形式发送请求，将一个函数名作为请求参数发给服务器，服务器将需要返回的数据封装在一段js代码中（把前端发来的函数名进行调用，把要发送的数据作为参数），然后把这段代码返回给前端，前端立刻执行这个函数。

3，代理服务器，将请求先发送给代理服务器，代理服务器装发给目标服务器。

22, 怎么列出一个文件夹下的所有文件？

可以使用nodeJS中的内置fs模块,调用fs.readDir读取文件夹中的所有文件和子文件夹, 传入递归函数,可读取子文件夹中的文件

23, Express中常用的中间件都有哪些？

static , bodyparser , session , multer ,

24, HTTP和HTTPS的区别是什么？

HTTP协议通常承载于TCP协议之上，在HTTP和TCP之间添加一个安全协议层（SSL或TSL），这个时候，就成了我们常说的HTTPS。

默认HTTP的端口号为80，HTTPS的端口号为443。

为什么HTTPS安全

因为网络请求需要中间有很多的服务器路由器的转发。中间的节点都可能篡改信息，而如果使用HTTPS，密钥在你和终点站才有。https之所以比http安全，是因为他利用ssl/tls协议传输。它包含证书，卸载，流量转发，负载均衡，页面适配，浏览器适配，refer传递等。保障了传输过程的安全性

25, Ajax中get和post两种请求方式的区别

（1）运行速度：get请求简单，运行速度也更快（存在缓存）；

（2）缓存：get存在缓存（优：提升速度，缺：不能及时获取最新数据）post没有缓存；

（3）数据储量：get有数据量的限制，post则没有限制

（4）数据安全：发送包含未知字符的用户输入时，post比get 更稳定也更可靠；

（5）传参方式：get参数拼接在url后，post放在send里面并且需要设置请求头xmr.setRequestHeader("content-type","application/x-www-form-urlencoded")

26, 什么是同源策略

同源策略是浏览器执行ajax请求时的一种安全策略, 它规定了浏览器只能请求端口、域名、协议全部相同的数据源, 而非同源的请求会被浏览器拦截

27, cookie和localstorage有什么区别？

都可以实现在用户的浏览器中存储一些数据。

不同：cookie是由服务端主导的，主要用于存储用户身份验证信息。localstorage是由前端js控制的，主要用于缓存业务逻辑数据。Cookie会随着请求头和响应头往返于服务器和浏览器之间。

28, localStorage、sessionStorage和cookie的区别

- 本地存储容量更大有5MB左右，cookie只有4KB

- 本地存储没有过期时间，localStorage持久保存，除非手动清除，sessionStorage窗口关闭自动清除

- cookie会在客户端与服务器端之间往返，服务器端可以操作cookie，本地存储只存储于本地

29, cookie 和session 的区别?

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。Session基于cookie。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

30, LocalStorage能存储多少数据量？如果数量过大会出现什么问题？怎么办？

一个域名下可存储5M的数据量, 存储的数据过大会导致部分数据丢失, 破坏数据结构

可优化数据结构,减小数据量

也可以使用window.postMessage实现跨源存储和读取,把数据存入其它域名下

31, 谈一下对token验证机制的理解

发起权限(登录)请求，后端验证通过生成token响应给客户端，客户端发送其他敏感请求，携带token，后端验证获取token识别用户身份

32, localstorage能否实现跨域访问？

正常情况下localstorage不能跨域访问，但是可以通过iframe实现间接的跨域访问，域名A的网站在页面中通过iframe打开域名B的页面，然后通过postMessage给B网页发送消息，域名B页面读取localstorage数据之后再通过postMessage将数据传递给A页面，从而实现跨域访问localStorage。也就是说B页面必须配合，A页面才能跨域访问。

33, webSocket是什么？适用于哪些网站？

webSocket是一种双工通信技术，可以实现服务器主动向客户端发送数据。

一般适用于需要实时通信的网站, 比如人工客服服务和在线页游等

34, webpack工具的功能是什么？

Webpack是为前端开发设计的自动化打包工具，能够对项目中的js、css、图片等资源进行打包（其中js可以直接打包，其他类型资源需要各自对应的loader支持），相比于传统的grunt、gulp等构建工具，webpack在打包js代码时，能够识别多种模块化语法，进行模块化打包。

Webpack还可以配合脚手架工具构建项目的框架

35, 什么是WebWorker？在哪些场景下需要使用WebWorker？

WebWorker是h5中新增的WebAPI，用于启动一个独立的线程，主线程和分线程只能通过相互发送消息进行通信。当前端页面中有耗时很长的代码需要执行时，可以放在worker中执行，否则会卡塞主线程，影响用户体验。

36, 你是否会使用混合(hybrid)APP开发技术进行手机APP开发？

开发手机app我一般使用uniapp去实现, 它不仅可以开发常规安卓和ios手机app还可以同时兼容发布各种小程序版本,开发效率比较高

37, ES6的async函数怎么使用？

Async函数没有返回值，或者只能返回Promise对象。在async函数的函数体中可以使用await，await 后面只能调用返回Promise的函数，且通过await调用之后返回值不再是promise，而是promise.then时所传递的数据（如果失败则会抛异常）。从而实现以同步代码的格式调用异步函数。

38, 在浏览器地址栏输入一个网址，从敲下回车键，到页面完全加载完毕，中间都发生了什么 ？

1，如果地址栏中输入的是一个域名，浏览器会先使用自己的DNS缓存进行域名解析，转为IP地址，如果缓存中不存在，则会请求使用上层DNS（操作系统的DNS），操作系统会先查询本地HOST文件，如果HOST文件中不存在则会使用网络设置的DNS进行域名解析。最终得到域名对应的IP地址。

2，获得IP地址后，浏览器按照IP地址进行连接(tcp连接)，连接成功之后按照http协议的格式（请求行，请求头，请求头）发送http请求，服务器会返回响应报文（响应行，响应头，响应体），浏览器收到响应报文后，会根据响应头中的Content-Type字段来决定如何进行下一步处理，对于普通的网页Content-Type值是text/html，浏览器就会在页面中打开本次请求响应体中的数据。

3，在打开页面时，浏览器首先对html文本进行解析，创建DOM树，然后将每个节点渲染到页面上，其中如果出现了附带资源的标签（例如img,script等），浏览器会再次按照这些资源的src发送请求，当请求完成之后将请求内容插入到页面中，其中script标签资源是同步加载的，其他资源是异步加载的。

解析为ip地址 向ip地址发送请求 获取请求先渲染html标签 在加载外部资源进行dom树的整合

39, 如何优化网页打开的速度？

1，尽量减少页面资源的请求次数（可以通过base64图片、合并图片、合并js，css文件实现）。

2，对页面代码进行压缩（主要是js代码压缩）

3，合理地使用懒加载。

4，对于不需要支持多国语言的中文网页，可以使用放弃使用UTF-8，使用GBK编码。

(以上是从前端角度进行优化，下面几条是从服务端角度优化)

5，网页中的静态资源使用CDN服务。

6，使用缓存，服务端使用redis进行接口缓存，同时在响应头中通过lastmodified,Etag等字段控制浏览器缓存。

7，使用gzip进行数据压缩。

8，使用多域名，部分浏览器对于同一个域名的并发请求量设置有上限，所以可以把页面资源分布在不同域名中，例如静态资源独自使用一个域名。

40, 什么是MVC和MVVM？

MVC:model-view-controller

MVVM:model-view-view-model，

MVC模式通过controller控制器协调model和view的交互，View 传送指令到 Controller，Controller 完成业务逻辑后，要求 Model 改变状态，Model 将新的数据发送到 View，用户得到反馈，所有通信都是单向的。

MVVM模式使用数据双向绑定，model和view直接进行交互。

41, 什么是XSS攻击？怎么防范？

Xss(cross-site scripting 跨站脚本攻击) 指的是攻击者往Web页面里插入恶意 html标签或者javascript代码。比如：攻击者在论坛中放一个看似安全的链接，骗取用户点击后，窃取cookie中的用户私密信息；或者攻击者在论坛中加一个恶意表单，当用户提交表单的时候，却把信息传送到攻击者的服务器中，而不是用户原本以为的信任站点。

防范：

首先代码里对用户输入的地方和变量都需要仔细检查长度和对<”,”>”,”;”,”’”等字符做过滤；其次任何内容写到页面之前都必须加以encode，避免不小心把html tag 弄出来。这一个层面做好，至少可以堵住超过一半的XSS 攻击。避免直接在cookie 中泄露用户隐私，例如email、密码等等。其次，通过使cookie 和系统ip 绑定来降低cookie 泄露后的危险。这样攻击者得到的cookie 没有实际价值，不可能拿来重放。如果网站不需要再浏览器端对cookie 进行操作，可以在Set-Cookie 末尾加上HttpOnly 来防止javascript 代码直接获取cookie 。

42, 什么是CSRF攻击？怎么防范？

Cross—Site Request Forgery，跨域请求伪造。

CSRF攻击攻击原理及过程如下：

1. 用户C打开浏览器，访问受信任网站A，输入用户名和密码请求登录网站A；

2.在用户信息通过验证后，网站A产生Cookie信息并返回给浏览器，此时用户登录网站A成功，可以正常发送请求到网站A；

3. 用户未退出网站A之前，在同一浏览器中，打开一个TAB页访问网站B；

4. 网站B接收到用户请求后，返回一些攻击性代码，并发出一个请求要求访问第三方站点A；

5. 浏览器在接收到这些攻击性代码后，根据网站B的请求，在用户不知情的情况下携带Cookie信息，向网站A发出请求。网站A并不知道该请求其实是由B发起的，所以会根据用户C的Cookie信息以C的权限处理该请求，导致来自网站B的恶意代码被执行。

防范：在服务端敏感接口中添加referer判断。