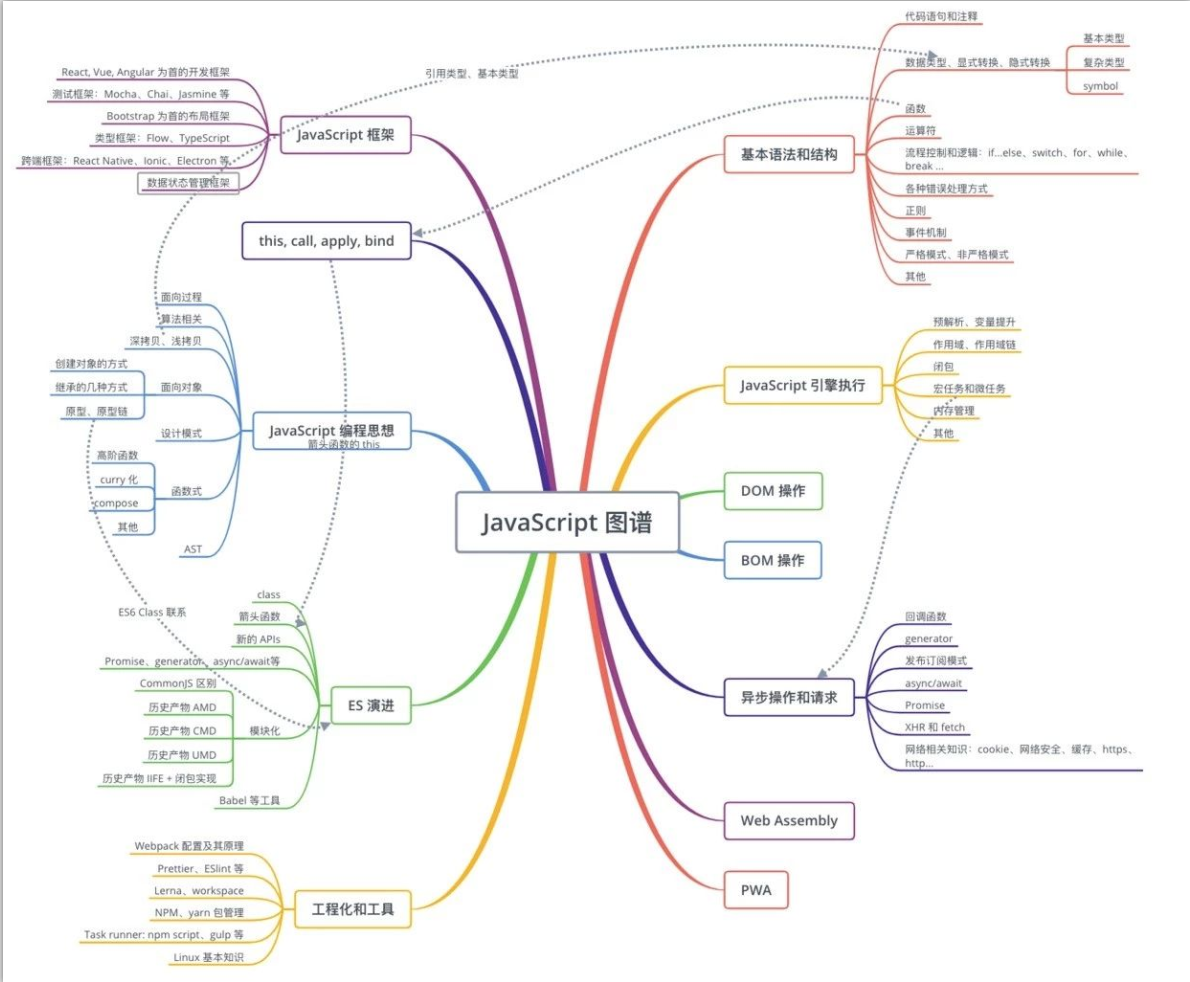
因为2020年4月份要找工作了，写文字太费时间了，还是这样比较快速的整理，还可以在手机上看，重点是背诵，希望，知道权重后，不要以为这只是为了消磨时间，时间被你消磨消磨就过去了呦。



一、js部分

1. **原型**

构造函数 ，是一种特殊的方法。主要用来在创建对象时初始化对象。每个构造函数都有prototype(原型)属性  
每个函数都有prototype(原型)属性，这个属性是一个指针，指向一个对象，  
这个对象的用途是包含特定类型的所有实例共享的属性和方法，即这个原型对象是用来给实例共享属性和方法的。  
而每个实例内部都有一个指向原型对象的指针。

1. **闭包**

简单来说就是函数嵌套函数，内部函数引用来外部函数的变量，从而导致来垃圾回收机制没有生效，变量被保存来下来。  
也就是所谓的内存泄漏，然后由于内存泄漏又会导致你项目逐渐变得卡顿等等问题。因此要避免内存泄漏。

我第一次解释闭包时，我常说函数中的函数;但是，它没有正确地描述闭包的确切含义。

闭包是在另一个作用域内创建一个封闭的词法范围。它通常会自动返回来生成这个词法环境。这个环境由创建闭包时在作用域内的任何局部变量组成。它就像一个微型工厂，用这些原料生产出具有特定功能的产品

*上学时：*

*函数嵌套形成闭包，保护里面的私有变量不被干扰，函数本身 以及他的外部作用域使用闭包可以形成独立的空间。*

*闭包的作用  
 1.防止变量名冲突  
 2.在闭包中对全局变量重新赋值，并且不影响全局变量  
 3.可以通过window.xx改变全局变量；  
 4.闭包可以用来封装；可以通过window.xxx=函数名；*

1. **原型链**

提到原型链就不得不提原型的继承，继承的完美实现方案是借助寄生组合继承，主要实现原理  
PersonB.prototype = Object.create(PersonA.prototype)实现来继承PersonA的原型  
当我们通过new关键字实例化的对象身上就有了PersonB自身的属性和方法，也有了PersonA的原型方法  
当实例化对象调用某个方法时会先在自身和原型上查找，然后是在\_proto\_上一层层查找，这种方式就是原型链。

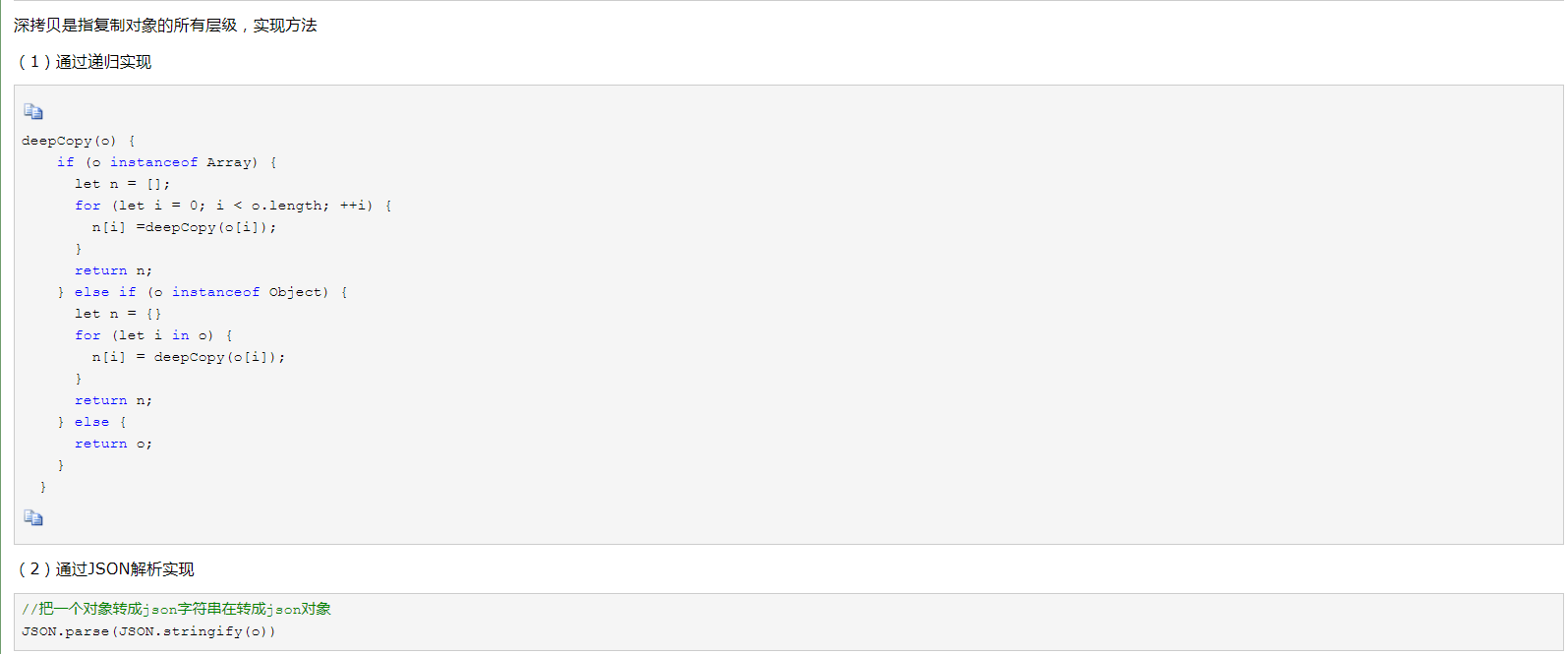
*上网查：*

*由于\_\_proto\_\_是任何对象都有的属性，而js里万物皆对线个，所以会形成一条\_\_proto\_\_连起来的链条， 递归访问\_\_proto\_\_必须最终到头，且值为null.  
  当js引擎查找对象的属性时，县查找对象本身是否存在该属性，如果不存在，会在原型链上查找，但不会查找自身的prototype*

1. **深拷贝浅拷贝**

深拷贝：  
通过利用JSON.parse(JSON.stringify(Object))来达到深拷贝的目的  
但是JSON深拷贝的缺点是undefined和function还有symbol类型是无法进行深拷贝的  
如有需要可以自己手动封装函数来达到目的  
浅拷贝：  
通过ES6新特性Object.assign()与扩展运算符来达到浅拷贝的目的





1. **陈述输入URL回车后的过程**

1.读取缓存：  
 搜索自身的 DNS 缓存。(如果 DNS 缓存中找到IP 地址就跳过了接下来查找 IP 地址步骤，直接访问该 IP 地址。)  
2.DNS 解析:将域名解析成 IP 地址  
3.TCP 连接：TCP 三次握手，简易描述三次握手  
 客户端：服务端你在么？  
 服务端：客户端我在，你要连接我么？  
 客户端：是的服务端，我要链接。  
 连接打通，可以开始请求来  
4.发送 HTTP 请求  
5.服务器处理请求并返回 HTTP 报文  
6.浏览器解析渲染页面  
7.断开连接：TCP 四次挥手  
  
关于第六步浏览器解析渲染页面又可以聊聊如果返回的是html页面  
根据 HTML 解析出 DOM 树

根据 CSS 解析生成 CSS 规则树  
结合 DOM 树和 CSS 规则树，生成渲染树  
根据渲染树计算每一个节点的信息  
根据计算好的信息绘制页面

1. **网页从输入网址到渲染完成经历了哪些过程？**

大致可以分为如下7步：

1. 输入网址；
2. 发送到DNS服务器，并获取域名对应的web服务器对应的ip地址；
3. 与web服务器建立TCP连接；
4. 浏览器向web服务器发送http请求；
5. web服务器响应请求，并返回指定url的数据（或错误信息，或重定向的新的url地址）；
6. 浏览器下载web服务器返回的数据及解析html源文件；
7. 生成DOM树，解析css和js，渲染页面，直至显示完成；
8. **陈述http**

基本概念：  
  
HTTP，全称为 HyperText Transfer Protocol，即为超文本传输协议。是互联网应用最为广泛的一种网络协议  
所有的 www 文件都必须遵守这个标准。  
  
http特性：  
  
HTTP 是无连接无状态的  
HTTP 一般构建于 TCP/IP 协议之上，默认端口号是 80  
HTTP 可以分为两个部分，即请求和响应。  
  
http请求：  
  
HTTP 定义了在与服务器交互的不同方式，最常用的方法有 4 种  
分别是 GET，POST，PUT， DELETE。URL 全称为资源描述符，可以这么认为：一个 URL 地址  
对应着一个网络上的资源，而 HTTP 中的 GET，POST，PUT，DELETE  
就对应着对这个资源的查询，修改，增添，删除4个操作。  
  
HTTP 请求由 3 个部分构成，分别是：状态行，请求头(Request Header)，请求正文。  
  
HTTP 响应由 3 个部分构成，分别是：状态行，响应头(Response Header)，响应正文。  
  
HTTP 响应中包含一个状态码，用来表示服务器对客户端响应的结果。  
状态码一般由3位构成：  
  
1xx : 表示请求已经接受了，继续处理。  
2xx : 表示请求已经处理掉了。  
3xx : 重定向。  
4xx : 一般表示客户端有错误，请求无法实现。  
5xx : 一般为服务器端的错误。  
  
比如常见的状态码：  
  
200 OK 客户端请求成功。  
301 Moved Permanently 请求永久重定向。  
302 Moved Temporarily 请求临时重定向。  
304 Not Modified 文件未修改，可以直接使用缓存的文件。  
400 Bad Request 由于客户端请求有语法错误，不能被服务器所理解。  
401 Unauthorized 请求未经授权，无法访问。  
403 Forbidden 服务器收到请求，但是拒绝提供服务。服务器通常会在响应正文中给出不提供服务的原因。  
404 Not Found 请求的资源不存在，比如输入了错误的URL。  
500 Internal Server Error 服务器发生不可预期的错误，导致无法完成客户端的请求。  
503 Service Unavailable 服务器当前不能够处理客户端的请求，在一段时间之后，服务器可能会恢复正常。  
  
大概还有一些关于hhtp请求和响应头信息的介绍。

1. **说一说js是什么语言**

js是一种运行在浏览器的脚本语言，这种语言主要的功能是可以制作出动态的页面的效果  
我们可以通过js+css+html布局来形成我们现在可以访问展示的页面  
  
js语言是弱语言类型， 因此我们在项目开发中当我们随意更改某个变量的数据类型后  
有可能会导致其他引用这个变量的方法中报错等等。

*上学时：*

*所有面向对象开发的语言就叫做编程语言，js是面向对象开发的语言  
JS主要分为三部分  
ECMAScript:语法 变量，命名，循环，数据类型，判断等等，流程控制语句。。。 完全兼容的  
DOM:document object model 文档对象模型 操作DOM的操作 不完全兼容--可以处理成兼容  
(其实就是各个html标签，节点等)  
BOM:browser object model 浏览器对象模型 操作window的操作 彻底不兼容，而且没法处理*

*JS在做UI效果时，就是用来操作css的；*

1. **说一说什么是跨域，怎么解决**

因为浏览器出于安全考虑，有同源策略。也就是说，如果协议、域名或者端口有一个不同就是跨域，Ajax 请求会失败。  
为来防止CSRF攻击  
1.JSONP  
 JSONP 的原理很简单，就是利用 <script> 标签没有跨域限制的漏洞。  
 通过 <script> 标签指向一个需要访问的地址并提供一个回调函数来接收数据当需要通讯时。  
 <script src="http://domain/api?param1=a&param2=b&callback=jsonp"></script>  
 <script>  
 function jsonp(data) {  
 console.log(data)  
 }  
 </script>  
 JSONP 使用简单且兼容性不错，但是只限于 get 请求。  
2.CORS  
 CORS 需要浏览器和后端同时支持。IE 8 和 9 需要通过 XDomainRequest 来实现。  
3.document.domain  
 该方式只能用于二级域名相同的情况下，比如 a.test.com 和 b.test.com 适用于该方式。  
  
 只需要给页面添加 document.domain = 'test.com' 表示二级域名都相同就可以实现跨域  
4.webpack配置proxyTable设置开发环境跨域  
5.nginx代理跨域  
6.iframe跨域  
7.postMessage  
 这种方式通常用于获取嵌入页面中的第三方页面数据。一个页面发送消息，另一个页面判断来源并接收消息

1. **说一说前端性能优化方案**

三个方面来说明前端性能优化  
一：webapck优化与开启gzip压缩  
 1.babel-loader用 include 或 exclude 来帮我们避免不必要的转译，不转译node\_moudules中的js文件  
 其次在缓存当前转译的js文件，设置loader: 'babel-loader?cacheDirectory=true'  
 2.文件采用按需加载等等  
 3.具体的做法非常简单，只需要你在你的 request headers 中加上这么一句：  
 accept-encoding:gzip  
 4.图片优化，采用svg图片或者字体图标  
 5.浏览器缓存机制，它又分为强缓存和协商缓存  
二：本地存储——从 Cookie 到 Web Storage、IndexedDB  
 说明一下SessionStorage和localStorage还有cookie的区别和优缺点  
三：代码优化  
 1.事件代理  
 2.事件的节流和防抖  
 3.页面的回流和重绘  
 4.EventLoop事件循环机制  
 5.代码优化等等

*答案二*

##### 1、减少 HTTP 请求数量

在浏览器与服务器进行通信时，主要是通过 HTTP 进行通信。浏览器与服务器需要经过三次握手，每次握手需要花费大量时间。而且不同浏览器对资源文件并发请求数量有限（不同浏览器允许并发数），一旦 HTTP 请求数量达到一定数量，资源请求就存在等待状态，这是很致命的，因此减少 HTTP 的请求数量可以很大程度上对网站性能进行优化。

**CSS Sprites**

国内俗称CSS精灵，这是将多张图片合并成一张图片达到减少HTTP请求的一种解决方案，可以通过CSS的background属性来访问图片内容。这种方案同时还可以减少图片总字节数。

**合并 CSS 和 JS 文件**

现在前端有很多工程化打包工具，如：grunt、gulp、webpack等。为了减少 HTTP 请求数量，可以通过这些工具再发布前将多个CSS或者多个JS合并成一个文件。

**采用 lazyLoad**

俗称懒加载，可以控制网页上的内容在一开始无需加载，不需要发请求，等到用户操作真正需要的时候立即加载出内容。这样就控制了网页资源一次性请求数量。

##### 2、控制资源文件加载优先级

浏览器在加载HTML内容时，是将HTML内容从上至下依次解析，解析到link或者script标签就会加载href或者src对应链接内容，为了第一时间展示页面给用户，就需要将CSS提前加载，不要受 JS 加载影响。

一般情况下都是CSS在头部，JS在底部。

##### 3、利用浏览器缓存

浏览器缓存是将网络资源存储在本地，等待下次请求该资源时，如果资源已经存在就不需要到服务器重新请求该资源，直接在本地读取该资源。

##### 4、减少重排（Reflow）

基本原理：重排是DOM的变化影响到了元素的几何属性（宽和高），浏览器会重新计算元素的几何属性，会使渲染树中受到影响的部分失效，浏览器会验证 DOM 树上的所有其它结点的visibility属性，这也是Reflow低效的原因。如果Reflow的过于频繁，CPU使用率就会急剧上升。

减少Reflow，如果需要在DOM操作时添加样式，尽量使用 增加class属性，而不是通过style操作样式。

##### 5、减少 DOM 操作

##### 6、图标使用 IconFont 替换

1. **说一说SessionStorage和localStorage还有cookie**

共同点：都是保存在浏览器端、且同源的  
不同点：  
 1.cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。  
 cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下  
 sessionStorage和localStorage不会自动把数据发送给服务器，仅在本地保存。  
 2.存储大小限制也不同，cookie数据不能超过4K，sessionStorage和localStorage可以达到5M  
 3.sessionStorage：仅在当前浏览器窗口关闭之前有效；  
 localStorage：始终有效，窗口或浏览器关闭也一直保存，本地存储，因此用作持久数据；  
 cookie：只在设置的cookie过期时间之前有效，即使窗口关闭或浏览器关闭  
 4.作用域不同  
 sessionStorage：不在不同的浏览器窗口中共享，即使是同一个页面；  
 localstorage：在所有同源窗口中都是共享的；也就是说只要浏览器不关闭，数据仍然存在  
 cookie: 也是在所有同源窗口中都是共享的.也就是说只要浏览器不关闭，数据仍然存在

1. **cookie的利弊？**

优点：极高的扩展性和可用性

1.通过良好的编程，控制保存在cookie中的session对象的大小。

2.通过加密和安全传输技术（SSL），减少cookie被破解的可能性。

3.只在cookie中存放不敏感数据，即使被盗也不会有重大损失。

4.控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。

缺点：

1.Cookie数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉。

2.安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。

3.有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

因为这些弊端（主要是cookie的大小和多少都受限制，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用），IE8以后，就出现一个web storage；它仅仅是为了本地缓存数据而存在；但是Cookie也是不可以或缺的：Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在。

1. **说说浏览器缓存**

缓存可以减少网络 IO 消耗，提高访问速度。浏览器缓存是一种操作简单、效果显著的前端性能优化手段  
很多时候，大家倾向于将浏览器缓存简单地理解为“HTTP 缓存”。  
但事实上，浏览器缓存机制有四个方面，它们按照获取资源时请求的优先级依次排列如下：  
  
Memory Cache  
Service Worker Cache  
HTTP Cache  
Push Cache  
  
缓存它又分为强缓存和协商缓存。优先级较高的是强缓存，在命中强缓存失败的情况下，才会走协商缓存  
 实现强缓存，过去我们一直用 expires。  
 当服务器返回响应时，在 Response Headers 中将过期时间写入 expires 字段，现在一般使用Cache-Control 两者同时出现使用Cache-Control  
   
 协商缓存，Last-Modified 是一个时间戳，如果我们启用了协商缓存，它会在首次请求时随着 Response Headers 返回：每次请求去判断这个时间戳是否发生变化。  
 从而去决定你是304读取缓存还是给你返回最新的数据

1. **如何理解 JS 中的this关键字？**

this表示当前对象，this的指向是根据调用的上下文来决定的，默认指向window对象。  
全局环境：全局环境就是在里面，这里的this始终指向的是window对象。  
局部环境：  
1.在全局作用域下直接调用函数，this指向window。  
2.对象函数调用，哪个对象调用就指向哪个对象。  
3.使用 new 实例化对象，在构造函数中的this指向实例化对象。  
4.使用call或apply改变this的指向。

1. **JavaScript 中的作用域（scope）是指什么？**

在 JavaScript 中，每个函数都有自己的作用域。作用域基本上是变量以及如何通过名称访问这些变量的规则的集合。只有函数中的代码才能访问函数作用域内的变量。

同一个作用域中的变量名必须是唯一的。一个作用域可以嵌套在另一个作用域内。如果一个作用域嵌套在另一个作用域内，最内部作用域内的代码可以访问另一个作用域的变量。

1. **解释 JavaScript 中的相等性。**

JavaScript 中有严格比较和类型转换比较：

严格比较（例如 ===）在不允许强制转型的情况下检查两个值是否相等；  
抽象比较（例如 ==）在允许强制转型的情况下检查两个值是否相等。

var a = "42";

var b = 42;

a == b; // true

a === b; // false

如果被比较的任何一个值可能是 true 或 false，要用 ===，而不是 ==；  
如果被比较的任何一个值是这些特定值（0、“”或 []），要用 ===，而不是 ==；  
在其他情况下，可以安全地使用 ==。它不仅安全，而且在很多情况下，它可以简化代码，并且提升代码可读性。

1. **解释一下变量的提升**

变量的提升是JavaScript的默认行为，这意味着将所有变量声明移动到当前作用域的顶部，并且可以在声明之前使用变量。初始化不会被提升，提升仅作用于变量的声明。

var x = 1

console.log(x + '——' + y) // 1——undefined

var y = 2

1. **如何理解事件委托**

在DOM树上绑定事件监听器并使用JS事件处理程序是处理客户端事件响应的典型方法。从理论上讲，我们可以将监听器附加到HTML中的任何DOM元素，但由于事件委派，这样做是浪费而且没必要的。

什么是事件委托？  
这是一种让父元素上的事件监听器也影响子元素的技巧。通常，事件传播（捕获和冒泡）允许我们实现事件委托。冒泡意味着当触发子元素（目标）时，也可以逐层触发该子元素的父元素，直到它碰到DOM绑定的原始监听器（当前目标）。捕获属性将事件阶段转换为捕获阶段，让事件下移到元素; 因此，触发方向与冒泡阶段相反。捕获的默认值为false。

1. **解释一下严格模式(strict mode)**

严格模式用于标准化正常的JavaScript语义。严格模式可以嵌入到非严格模式中，关键字 ‘use strict’。使用严格模式后的代码应遵循JS严格的语法规则。例如，分号在每个语句声明之后使用。

1. **解释 JavaScript 中的 null 和 undefined。**

JavaScript 中有两种底层类型：null 和 undefined。它们代表了不同的含义：

尚未初始化：undefined；

空值：null。

//null和undefined是两个不同的对象 null == null //true null === null //true null == undefined //true null === undefined //flase

1. **解释事件冒泡以及如何阻止它？**

事件冒泡是指嵌套最深的元素触发一个事件，然后这个事件顺着嵌套顺序在父元素上触发。

防止事件冒泡的一种方法是使用 event.cancelBubble 或 event.stopPropagation()（低于 IE 9）。

1. **window.onload的作用？**

window.onload是等文档和资源都加载完成后调用的事件，保证js获取元素 的时候，已经加载。

1. **说说你对作用域链的理解？**

作用域链是js中的一种查找机制，从当前作用域查找，当前作用域没有往上一级作用域查找，一直到最外层，如果都找不到则是is not define

1. **定时器的分类？他们的区别及用法是什么？**

Js中有两种定时器：setInterval：间歇执行，setTimeout：延迟执行

setInterval(函数，时间) 时间单位ms，

语法：setInterval(function(){},1000); 隔一秒执行函数一次

一般用于倒计时，轮播图

setTimeout(函数，时间) 时间单位ms，

语法：setTimeout(function(){},1000); 延迟一秒执行函数一次，只会执行一次

一般用于广告，广告弹出层

1. **split() join() 的区别？**

split是字符串的方法，将字符串按照特定标志分割成数组  
例：“u&s”.split(“&”) ------ [“u”,”s”]  
Join:是数组方法，将数组按标志组合成字符串 [“u”,”s”].join（“-”）–‘u-s’

1. **什么是window对象? 什么是document对象?**

window是js中最大的对象，表示窗口，包含document  
document文档对象，表示HTML

1. **offsetWidth、clientWidth、scrollTop的区别？**

offsetWidth：占位宽，包含 内容宽+左右padding+左右border  
clientWidth：可视宽，包含 内容宽+左右padding  
scrollTop：页面被卷去的高

1. **事件、IE与火狐的事件机制有什么区别？**

IE的事件流是冒泡流，而火狐同时支持冒泡流和捕获流。

1. **事件绑定和普通事件有什么区别。**

标签.事件：如果给同一个元素添加同一个事件，后面的会覆盖前面  
事件绑定：可以给同一个元素添加同一个事件，不会被覆盖

1. **解释一下事件流？**

事件捕获阶段：当事件发生的时候，将事件从window依次往子元素传递  
确定目标阶段：确定事件目标  
事件冒泡阶段：事件目标开始处理事件，处理完以后会将事件依次传递给父元素，一直到window  
事件都是在事件冒泡处理,ie只有冒泡

1. **call()和apply()的区别和作用？**

都是用来改变this的指向，在传值的过程中会有些区别  
call（this的指向，参数1，参数2…）  
apply（this的指向,[参数1，参数2…]）

1. **ajax请求时，如何解释json数据。**

如果是字符串形式的json：eval("("+ajax.response+")")  
如果是本地的json文件：JSON.parse(data)

1. **同步和异步的区别?**

异步：客户端与服务器请求数据的过程中，可以做其他的事情  
同步：客户端与服务器请求数据的过程中，不能做其他的事情

1. **js哪些操作会造成内存泄露？**

1）意外的全局变量引起的内存泄露  
function leak(){ leak=“xxx”;//leak成为一个全局变量，不会被回收 }  
2）被遗忘的定时器或者回调  
3）闭包引起的内存泄漏

1. **$(document).ready() 方法和window.onload 有什么区别？**

window.onload与ready的区别  
1.window.onload会覆盖之前的  
ready的区别，不会覆盖，会叠加  
2.window.onload等文档和资源都加载完成以后调用  
ready只要文档加载完成以后就会调用

二、vue部分

1. **vuex**

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态  
并以相应的规则保证状态以一种可预测的方式发生变化。  
tate：Vuex 使用单一状态树——是的，用一个对象就包含了全部的应用层级状态。  
mutation：更改 Vuex 的 store 中的状态的唯一方法是提交 mutation  
action: action 提交的是 mutation，而不是直接变更状态。action 可以包含任意异步操作。  
getter: 相当于Vue中的computed计算属性

1. **vue-router**

Vue Router 是 Vue.js 官方的路由管理器。它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌  
<router-link>和<router-view>和<keep-alive>

1. **Vue通信**

第一种：props和$emit  
第二种：中央事件总线 EventBus(基本不用)  
第三种：vuex（状态管理器）  
第四种：$parent 和 $children  
当然还有其他办法，但是基本不常用

1. **说说Vue原理**

Vue是采用数据劫持配合发布者-订阅者模式，通过Object.defineProperty来()来劫持各个属性的getter和setter  
在数据发生变化的时候，发布消息给依赖收集器，去通知观察者，做出对应的回调函数去更新视图。  
  
具体就是：  
MVVM作为绑定的入口，整合Observe,Compil和Watcher三者，通过Observe来监听model的变化  
通过Compil来解析编译模版指令，最终利用Watcher搭起Observe和Compil之前的通信桥梁  
从而达到数据变化 => 更新视图，视图交互变化(input) => 数据model变更的双向绑定效果。

1. **Vue路由守卫有哪些，怎么设置，使用场景等**

常用的两个路由守卫：router.beforeEach 和 router.afterEach  
  
每个守卫方法接收三个参数：  
  
to: Route: 即将要进入的目标 路由对象  
  
from: Route: 当前导航正要离开的路由  
  
next: Function: 一定要调用该方法来 resolve 这个钩子。  
  
在项目中，一般在beforeEach这个钩子函数中进行路由跳转的一些信息判断。  
判断是否登录，是否拿到对应的路由权限等等。

1. **谈谈你对MVVM开发模式的理解**

MVVM分为Model、View、ViewModel三者。

Model：代表数据模型，数据和业务逻辑都在Model层中定义；

View：代表UI视图，负责数据的展示；

ViewModel：负责监听Model中数据的改变并且控制视图的更新，处理用户交互操作；

Model和View并无直接关联，而是通过ViewModel来进行联系的，Model和ViewModel之间有着双向数据绑定的联系。因此当Model中的数据改变时会触发View层的刷新，View中由于用户交互操作而改变的数据也会在Model中同步。

这种模式实现了Model和View的数据自动同步，因此开发者只需要专注对数据的维护操作即可，而不需要自己操作dom。

1. **Vue 有哪些指令？**

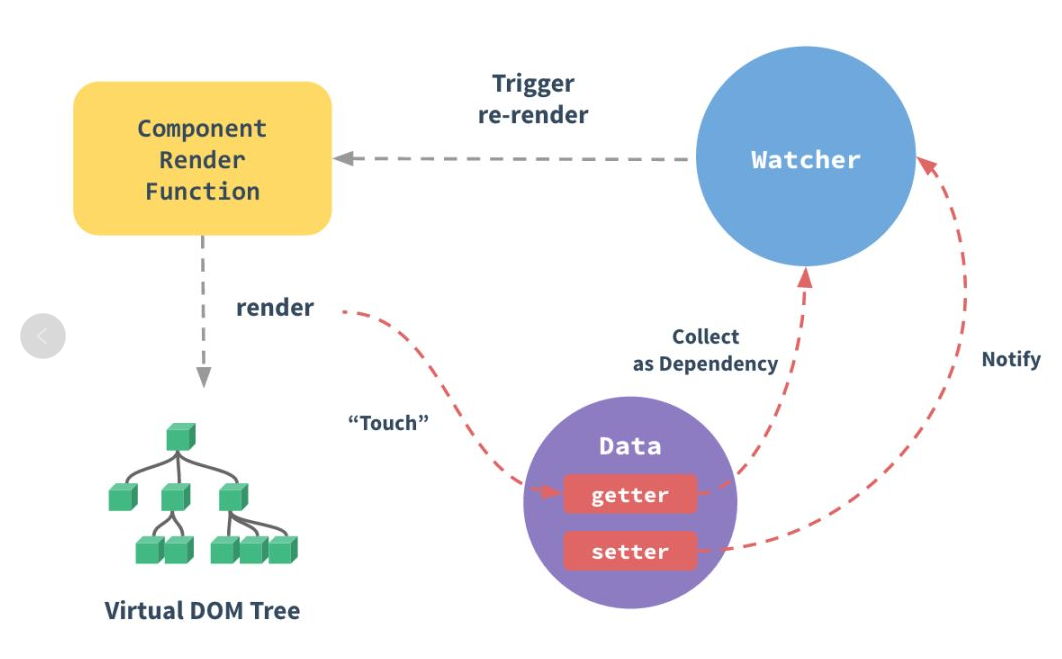
v-html、v-show、v-if、v-for等等。

1. **v-if 和 v-show 有什么区别？**

v-show 仅仅控制元素的显示方式，将 display 属性在 block 和 none 来回切换；而v-if会控制这个 DOM 节点的存在与否。当我们需要经常切换某个元素的显示/隐藏时，使用v-show会更加节省性能上的开销；当只需要一次显示或隐藏时，使用v-if更加合理。

1. **简述Vue的响应式原理**

当一个Vue实例创建时，vue会遍历data选项的属性，用 Object.defineProperty 将它们转为getter/setter并且在内部追踪相关依赖，在属性被访问和修改时通知变化。 每个组件实例都有相应的watcher程序实例，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的setter被调用时，会通知watcher重新计算，从而致使它关联的组件得以更新。



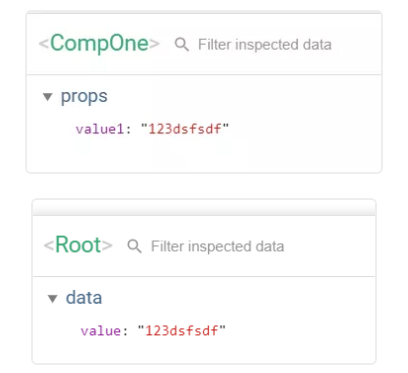
1. **Vue中如何在组件内部实现一个双向数据绑定？**

假设有一个输入框组件，用户输入时，同步父组件页面中的数据。

具体思路：父组件通过props传值给子组件，子组件通过 $emit 来通知父组件修改相应的props值，具体实现如下：



可以看到，当输入数据时，父子组件中的数据是同步改变的：





1. **Vue中如何监控某个属性值的变化？**



1. **Vue中给data中的对象属性添加一个新的属性时会发生什么，如何解决？**





1. **delete和Vue.delete删除数组的区别**



1. **如何优化SPA应用的首屏加载速度慢的问题？**

将公用的JS库通过script标签外部引入，减小 app.bundel 的大小，让浏览器并行下载资源文件，提高下载速度；

在配置 路由时，页面和组件使用懒加载的方式引入，进一步缩小 app.bundel 的体积，在调用某个组件时再加载对应的js文件；

加一个首屏loading图，提升用户体验；

1. **如何理解MVVM原理**
2. **响应式数据的原理是什么**
3. **vue中是如何检测数组变化**
4. **为何vue采用异步渲染**
5. **nextTick实现原理**
6. **vue组件的生命周期**
7. **Ajax请求放在哪个生命周期中**
8. **何时需要使用beforeDestroy？**
9. **vue父子组件生命周期调用顺序**
10. **vue中Computed的特点**
11. **Watch中的deep：true是如何实现的**
12. **Vue中事件绑定的原理**
13. **vue中v-html会导致那些问题**
14. **vue中v-if和v-show的区别**
15. **为什么v-for和v-if不能连用**
16. **v-model中的实现原理以及如何自定义v-model**
17. **组件中的data为什么是一个函数**
18. **vue组件如何通信**
19. **什么是作用域插槽**
20. **用vnode来描述一个dom结构**
21. **diff算法的时间复杂度**
22. **简书vue中diff算法原理**
23. **v-for中为什么要用key**
24. **描述组件渲染和更新过程**
25. **vue中模板编译原理**
26. **vue 中常见性能优化**
27. **vue中相同逻辑如何抽离**
28. **为什么要使用异步组件**
29. **谈谈你对keep-alive的了解**
30. **实现hash路由和history路由**
31. **vue-router中导航守卫有哪些**
32. **action和mutation区别**
33. **简述vuex工作原理**
34. **vue3.0你知道有哪些改进**

三、webpack部分

1. **webpack配置入口出口**

module.exports={  
 //入口文件的配置项  
 entry:{},  
 //出口文件的配置项  
 output:{},  
 //模块：例如解读CSS,图片如何转换，压缩  
 module:{},  
 //插件，用于生产模版和各项功能  
 plugins:[],  
 //配置webpack开发服务功能  
 devServer:{}  
}  
简单描述了一下这几个属性是干什么的。

1. **webpack3和webpack4区别**

1.mode  
  
webpack增加了一个mode配置，只有两种值development | production。对不同的环境他会启用不同的配置。  
  
2.CommonsChunkPlugin  
  
CommonChunksPlugin已经从webpack4中移除。  
可使用optimization.splitChunks进行模块划分（提取公用代码）。  
但是需要注意一个问题，默认配置只会对异步请求的模块进行提取拆分，如果要对entry进行拆分  
需要设置optimization.splitChunks.chunks = 'all'。  
  
3.webpack4使用MiniCssExtractPlugin取代ExtractTextWebpackPlugin。  
  
4.代码分割。  
  
使用动态import，而不是用system.import或者require.ensure  
  
5.vue-loader。  
  
使用vue-loader插件为.vue文件中的各部分使用相对应的loader，比如css-loader等  
  
6.UglifyJsPlugin  
  
现在也不需要使用这个plugin了，只需要使用optimization.minimize为true就行，production mode下面自动为true  
  
optimization.minimizer可以配置你自己的压缩程序

四、工作问题

1. **你在工作中遇到那些问题，解决方法是什么**

经常遇到的问题就是Cannot read property ‘prototype’ of undefined  
解决办法通过浏览器报错提示代码定位问题，解决问题  
  
Vue项目中遇到视图不更新，方法不执行，埋点不触发等问题  
一般解决方案查看浏览器报错，查看代码运行到那个阶段未之行结束，阅读源码以及相关文档等  
然后举出来最近开发的项目中遇到的算是两个比较大的问题。

1. **说一说你用过的css布局**

gird布局，layout布局，flex布局，双飞翼，圣杯布局等

1. **说一说你用过的UI框架**

Element-UI Vant

五、ES6

1. **ES6新特性**
2. ES6引入来严格模式  
    变量必须声明后在使用  
    函数的参数不能有同名属性, 否则报错  
    不能使用with语句 (说实话我基本没用过)  
    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 (说实话我基本没用过)  
    不能删除不可删除的数据, 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.caller (说实话我基本没用过)  
    不能使用arguments.callee (说实话我基本没用过)  
    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 (说实话我基本没用过)  
    增加了保留字（比如protected、static和interface）  
     
   2.关于let和const新增的变量声明  
     
   3.变量的解构赋值  
     
   4.字符串的扩展  
    includes()：返回布尔值，表示是否找到了参数字符串。  
    startsWith()：返回布尔值，表示参数字符串是否在原字符串的头部。  
    endsWith()：返回布尔值，表示参数字符串是否在原字符串的尾部。  
   5.数值的扩展  
    Number.isFinite()用来检查一个数值是否为有限的（finite）。  
    Number.isNaN()用来检查一个值是否为NaN。  
   6.函数的扩展  
    函数参数指定默认值  
   7.数组的扩展  
    扩展运算符  
   8.对象的扩展  
    对象的解构  
   9.新增symbol数据类型  
     
   10.Set 和 Map 数据结构  
    ES6 提供了新的数据结构 Set。它类似于数组，但是成员的值都是唯一的，没有重复的值。Set 本身是一个构造函数，用来生成 Set 数据结构。  
      
    Map它类似于对象，也是键值对的集合，但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键。  
   11.Proxy  
    Proxy 可以理解成，在目标对象之前架设一层“拦截”，外界对该对象的访问  
    都必须先通过这层拦截，因此提供了一种机制，可以对外界的访问进行过滤和改写。  
    Proxy 这个词的原意是代理，用在这里表示由它来“代理”某些操作，可以译为“代理器”。  
    Vue3.0使用了proxy  
   12.Promise  
    Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。  
    特点是：  
    对象的状态不受外界影响。  
    一旦状态改变，就不会再变，任何时候都可以得到这个结果。  
   13.async 函数  
    async函数对 Generator 函数的区别：  
    （1）内置执行器。  
    Generator 函数的执行必须靠执行器，而async函数自带执行器。也就是说，async函数的执行，与普通函数一模一样，只要一行。  
    （2）更好的语义。  
    async和await，比起星号和yield，语义更清楚了。async表示函数里有异步操作，await表示紧跟在后面的表达式需要等待结果。  
    （3）正常情况下，await命令后面是一个 Promise 对象。如果不是，会被转成一个立即resolve的 Promise 对象。  
    （4）返回值是 Promise。  
    async函数的返回值是 Promise 对象，这比 Generator 函数的返回值是 Iterator 对象方便多了。你可以用then方法指定下一步的操作。  
   14.Class  
    class跟let、const一样：不存在变量提升、不能重复声明...  
    ES6 的class可以看作只是一个语法糖，它的绝大部分功能  
    ES5 都可以做到，新的class写法只是让对象原型的写法更加清晰、更像面向对象编程的语法而已。  
   15.Module  
    ES6 的模块自动采用严格模式，不管你有没有在模块头部加上"use strict";。  
    import和export命令以及export和export default的区别
3. **Set,Map解构**

ES6 提供了新的数据结构 Set。  
它类似于数组，但是成员的值都是唯一的，没有重复的值。Set 本身是一个构造函数，用来生成 Set 数据结构。  
  
ES6 提供了 Map 数据结构。它类似于对象，也是键值对的集合，但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键。

1. **Promise是什么，解决了什么，之前怎么实现的**

Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。  
 解决来之前在请求中回调请求产生的回调地狱，使得现在的代码更加合理更加优雅，也更加容易定位查找问题。

六、jquery问题

1. **jQuery获取的dom对象和原生的dom对象有何区别？**

js原生获取的dom是一个对象，jQuery对象就是一个数组对象，其实就是选择出来的元素的数组集合，所以说他们两者是不同的对象类型不等价。



1. **jQuery如何扩展自定义方法**

