**如何实现浏览器内多个标签页之间的通信?**  
调用localstorge、cookies等本地存储方式  
**线程与进程的区别**  
一个程序至少有一个进程,一个进程至少有一个线程。  
线程的划分尺度小于进程，使得多线程程序的并发性高。   
另外，进程在执行过程中拥有独立的内存**单元，而多个**线程共享内存，从而极大地提高了程序的运行效率。   
线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。   
从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。  
但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。  
**你如何对网站的文件和资源进行优化？**  
期待的解决方案包括：  
文件合并  
文件最小化/文件压缩  
使用 CDN 托管  
缓存的使用（多个域名来提供缓存）  
其他

**请说出三种减少页面加载时间的方法。**  
1.优化图片   
2.图像格式的选择（GIF：提供的颜色较少，可用在一些对颜色要求不高的地方）   
3.优化CSS（压缩合并css，如margin-top,margin-left...)   
4.网址后加斜杠（如www.campr.com/目录，会判断这个“目录是什么文件类型，或者是目录。）   
5.标明高度和宽度（如果浏览器没有找到这两个参数，它需要一边下载图片一边计算大小，如果图片很多，浏览器需要不断地调整页面。这不但影响速度，也影响浏览体验。   
当浏览器知道了高度和宽度参数后，即使图片暂时无法显示，页面上也会腾出图片的空位，然后继续加载后面的内容。从而加载时间快了，浏览体验也更好了。）   
6.减少http请求（合并文件，合并图片）。

**你都使用哪些工具来测试代码的性能？**  
Profiler, JSPerf（http://jsperf.com/nexttick-vs-setzerotimeout-vs-settimeout）, Dromaeo  
什么是 FOUC（无样式内容闪烁）？你如何来避免 FOUC？  
FOUC - Flash Of Unstyled Content 文档样式闪烁  
<style type="text/css" media="all">@import "../fouc.css";</style>   
而引用CSS文件的@import就是造成这个问题的罪魁祸首。IE会先加载整个HTML文档的DOM，然后再去导入外部的CSS文件，因此，在页面DOM加载完成到CSS导入完成中间会有一段时间页面上的内容是没有样式的，这段时间的长短跟网速，电脑速度都有关系。  
解决方法简单的出奇，只要在<head>之间加入一个<link>或者<script>元素就可以了。

**null和undefined的区别？**  
null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。  
当声明的变量还未被初始化时，变量的默认值为undefined。 null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。  
undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。典型用法是：  
（1）变量被声明了，但没有赋值时，就等于undefined。  
（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。  
（3）对象没有赋值的属性，该属性的值为undefined。  
（4）函数没有返回值时，默认返回undefined。

null表示"没有对象"，即该处不应该有值。典型用法是：  
（1） 作为函数的参数，表示该函数的参数不是对象。  
（2） 作为对象原型链的终点。

**new操作符具体干了什么呢?**  
   1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。  
   2、属性和方法被加入到 this 引用的对象中。  
   3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。  
var obj  = {};  
obj.\_\_proto\_\_ = Base.prototype;  
Base.call(obj);

**js延迟加载的方式有哪些？**  
defer和async、动态创建DOM方式（创建script，插入到DOM中，加载完毕后callBack）、按需异步载入js  
如何解决跨域问题?  
jsonp、 document.domain+iframe、window.name、window.postMessage、服务器上设置代理页面  
jsonp的原理是动态插入script标签  
具体参见：详解js跨域问题（https://segmentfault.com/a/1190000000718840）  
**documen.write和 innerHTML的区别**  
document.write只能重绘整个页面  
innerHTML可以重绘页面的一部分  
.call() 和 .apply() 的区别和作用？  
作用：动态改变某个类的某个方法的运行环境。 区别参见：JavaScript学习总结（四）function函数部分（https://segmentfault.com/a/1190000000660786#articleHeader15）  
**哪些操作会造成内存泄漏？**  
内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。  
垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。  
setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。  
闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）  
详见：详解js变量、作用域及内存（https://segmentfault.com/a/1190000000687844）  
**JavaScript中的作用域与变量声明提升？**  
详见：详解JavaScript函数模式（https://segmentfault.com/a/1190000000758184#articleHeader5）  
  
**如何判断当前脚本运行在浏览器还是node环境中？  
  
通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中  
其他问题？  
  
你遇到过比较难的技术问题是？你是如何解决的？  
  
列举IE 与其他浏览器不一样的特性？  
  
什么叫优雅降级和渐进增强？**  
优雅降级：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于IE独特的盒模型布局问题，针对不同版本的IE的hack实践过优雅降级了,为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效.  
渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能,向页面增加无害于基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。  
详见：css学习归纳总结（一）（https://segmentfault.com/a/1190000000800711）  
**WEB应用从服务器主动推送Data到客户端有那些方式？**  
Javascript数据推送  
Commet：基于HTTP长连接的服务器推送技术  
基于WebSocket的推送方案  
SSE（Server-Send Event）：服务器推送数据新方式

**对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？**  
前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。  
    1、实现界面交互  
    2、提升用户体验  
    3、有了Node.js，前端可以实现服务端的一些事情前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，  
参与项目，快速高质量完成实现效果图，精确到1px；  
与团队成员，UI设计，产品经理的沟通；  
做好的页面结构，页面重构和用户体验；  
处理hack，兼容、写出优美的代码格式；  
针对服务器的优化、拥抱最新前端技术。

**你有哪些性能优化的方法？**  
（详情请看雅虎14条性能优化原则）（https://segmentfault.com/blog/trigkit4/1190000000656717）。  
  （1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。  
  （2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数  
  （3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。  
  （4） 当需要设置的样式很多时设置className而不是直接操作style。  
  （5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。  
  （6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。  
  （7） 图片预加载，将样式表放在顶部，将脚本放在底部  加上时间戳。  
详情：http://segmentfault.com/blog/trigkit4/1190000000691919  
**一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？**  
分为4个步骤：  
   （1），当发送一个URL请求时，不管这个URL是Web页面的URL还是Web页面上每个资源的URL，浏览器都会开启一个线程来处理这个请求，同时在远程DNS服务器上启动一个DNS查询。这能使浏览器获得请求对应的IP地址。  
   （2）， 浏览器与远程Web服务器通过TCP三次握手协商来建立一个TCP/IP连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在 浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。  
   （3），一旦TCP/IP连接建立，浏览器会通过该连接向远程服务器发送HTTP的GET请求。远程服务器找到资源并使用HTTP响应返回该资源，值为200的HTTP响应状态表示一个正确的响应。  
   （4），此时，Web服务器提供资源服务，客户端开始下载资源。  
请求返回后，便进入了我们关注的前端模块  
简单来说，浏览器会解析HTML生成DOM Tree，其次会根据CSS生成CSS Rule Tree，而javascript又可以根据DOM API操作DOM  
详情：从输入 URL 到浏览器接收的过程中发生了什么事情？（https://segmentfault.com/a/1190000000697254）  
**平时如何管理你的项目？**  
先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；  
编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；  
标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；  
页面进行标注（例如 页面 模块 开始和结束）；  
CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；  
JS 分文件夹存放 命名以该JS功能为准的英文翻译。  
图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理   
**说说最近最流行的一些东西吧？常去哪些网站？**  
Node.js、Mongodb、npm、MVVM、MEAN、three.js,React 。  
网站：w3cfuns,sf,hacknews,CSDN,慕课，博客园，InfoQ,w3cplus等

**javascript对象的几种创建方式**  
1，工厂模式  
2，构造函数模式  
3，原型模式  
4，混合构造函数和原型模式  
5，动态原型模式  
6，寄生构造函数模式  
7，稳妥构造函数模式  
**javascript继承的6种方法**  
1，原型链继承  
2，借用构造函数继承  
3，组合继承(原型+借用构造)  
4，原型式继承  
5，寄生式继承  
6，寄生组合式继承  
详情：JavaScript继承方式详解（https://segmentfault.com/a/1190000002440502）  
**ajax过程**  
(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象.  
(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息.  
(3)设置响应HTTP请求状态变化的函数.  
(4)发送HTTP请求.  
(5)获取异步调用返回的数据.  
(6)使用JavaScript和DOM实现局部刷新.  
详情：JavaScript学习总结（七）Ajax和Http状态字（https://segmentfault.com/a/1190000000691919）  
**异步加载和延迟加载**  
1.异步加载的方案： 动态插入script标签  
2.通过ajax去获取js代码，然后通过eval执行  
3.script标签上添加defer或者async属性  
4.创建并插入iframe，让它异步执行js  
5.延迟加载：有些 js 代码并不是页面初始化的时候就立刻需要的，而稍后的某些情况才需要的。  
前端安全问题？  
**sql注入原理**  
就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。  
总的来说有以下几点：  
1.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。  
2.永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。  
3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。  
4.不要把机密信息明文存放，请加密或者hash掉密码和敏感的信息。  
**XSS原理及防范**  
Xss(cross-site scripting)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。  
XSS防范方法  
1.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。   
2.避免直接在cookie 中泄露用户隐私，例如email、密码等等。

3.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。   
4.尽量采用POST 而非GET 提交表单  
**XSS与CSRF有什么区别吗？**  
XSS是获取信息，不需要提前知道其他用户页面的代码和数据包。CSRF是代替用户完成指定的动作，需要知道其他用户页面的代码和数据包。  
要完成一次CSRF攻击，受害者必须依次完成两个步骤：  
登录受信任网站A，并在本地生成Cookie。 　　  
在不登出A的情况下，访问危险网站B。  
CSRF的防御  
服务端的CSRF方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数。   
使用验证码  
ie各版本和chrome可以并行下载多少个资源  
IE6 两个并发，iE7升级之后的6个并发，之后版本也是6个  
Firefox，chrome也是6个  
**javascript里面的继承怎么实现，如何避免原型链上面的对象共享**  
用构造函数和原型链的混合模式去实现继承，避免对象共享可以参考经典的extend()函数，很多前端框架都有封装的，就是用一个空函数当做中间变量  
grunt， YUI compressor 和 google clojure用来进行代码压缩的用法。  
YUI Compressor 是一个用来压缩 JS 和 CSS 文件的工具，采用Java开发。  
使用方法：  
//压缩JS  
java -jar yuicompressor-2.4.2.jar --type js --charset utf-8 -v src.js > packed.js  
//压缩CSS  
java -jar yuicompressor-2.4.2.jar --type css --charset utf-8 -v src.css > packed.css  
详情请见：你需要掌握的前端代码性能优化工具（https://segmentfault.com/a/1190000002585760）  
**Flash、Ajax各自的优缺点，在使用中如何取舍？**  
1、Flash ajax对比  
Flash适合处理多媒体、矢量图形、访问机器；对CSS、处理文本上不足，不容易被搜索。  
Ajax对CSS、文本支持很好，支持搜索；多媒体、矢量图形、机器访问不足。  
共同点：与服务器的无刷新传递消息、用户离线和在线状态、操作DOM  
**请解释一下 JavaScript 的同源策略。**  
概念:同源策略是客户端脚本（尤其是Javascript）的重要的安全度量标准。它最早出自Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。  
这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议。 指一段脚本只能读取来自同一来源的窗口和文档的属性。  
为什么要有同源限制？  
我们举例说明：比如一个黑客程序，他利用Iframe把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过Javascript读取到你的表单中input中的内容，这样用户名，密码就轻松到手了。  
什么是 "use strict"; ? 使用它的好处和坏处分别是什么？  
ECMAscript 5添加了第二种运行模式："严格模式"（strict mode）。顾名思义，这种模式使得Javascript在更严格的条件下运行。  
设立"严格模式"的目的，主要有以下几个：  
- 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;  
- 消除代码运行的一些不安全之处，保证代码运行的安全；  
- 提高编译器效率，增加运行速度；  
- 为未来新版本的Javascript做好铺垫。  
注：经过测试IE6,7,8,9均不支持严格模式。  
缺点： 现在网站的JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。  
**GET和POST的区别，何时使用POST？**  
GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符  
POST：一般用于修改服务器上的资源，对所发送的信息没有限制。  
GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值，  
也就是说Get是通过地址栏来传值，而Post是通过提交表单来传值。  
然而，在以下情况中，请使用 POST 请求：  
无法使用缓存文件（更新服务器上的文件或数据库）  
向服务器发送大量数据（POST 没有数据量限制）  
发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠  
**哪些地方会出现css阻塞，哪些地方会出现js阻塞？**  
js的阻塞特性：所有浏览器在下载JS的时候，会阻止一切其他活动，比如其他资源的下载，内容的呈现等等。直到JS下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。为了提高用户体验，新一代浏览器都支持并行下载JS，但是JS下载仍然会阻塞其它资源的下载（例如.图片，css文件等）。  
由于浏览器为了防止出现JS修改DOM树，需要重新构建DOM树的情况，所以就会阻塞其他的下载和呈现。  
嵌入JS会阻塞所有内容的呈现，而外部JS只会阻塞其后内容的显示，2种方式都会阻塞其后资源的下载。也就是说外部样式不会阻塞外部脚本的加载，但会阻塞外部脚本的执行。  
CSS怎么会阻塞加载了？CSS本来是可以并行下载的，在什么情况下会出现阻塞加载了(在测试观察中，IE6下CSS都是阻塞加载）  
当CSS后面跟着嵌入的JS的时候，该CSS就会出现阻塞后面资源下载的情况。而当把嵌入JS放到CSS前面，就不会出现阻塞的情况了。  
根本原因：因为浏览器会维持html中css和js的顺序，样式表必须在嵌入的JS执行前先加载、解析完。而嵌入的JS会阻塞后面的资源加载，所以就会出现上面CSS阻塞下载的情况。  
**嵌入JS应该放在什么位置？**  
   1、放在底部，虽然放在底部照样会阻塞所有呈现，但不会阻塞资源下载。  
   2、如果嵌入JS放在head中，请把嵌入JS放在CSS头部。  
   3、使用defer（只支持IE）  
   4、不要在嵌入的JS中调用运行时间较长的函数，如果一定要用，可以用`setTimeout`来调用  
Javascript无阻塞加载具体方式  
将脚本放在底部。<link>还是放在head中，用以保证在js加载前，能加载出正常显示的页面。<script>标签放在</body>前。  
成组脚本：由于每个<script>标签下载时阻塞页面解析过程，所以限制页面的<script>总数也可以改善性能。适用于内联脚本和外部脚本。  
非阻塞脚本：等页面完成加载后，再加载js代码。也就是，在window.onload事件发出后开始下载代码。 （1）defer属性：支持IE4和fierfox3.5更高版本浏览器 （2）动态脚本元素：文档对象模型（DOM）允许你使用js动态创建HTML的几乎全部文档内容。代码如下：  
<script>  
var script=document.createElement("script");  
script.type="text/javascript";  
script.src="file.js";  
document.getElementsByTagName("head")[0].appendChild(script);  
</script>  
此技术的重点在于：无论在何处启动下载，文件额下载和运行都不会阻塞其他页面处理过程。即使在head里（除了用于下载文件的http链接）。  
**闭包相关问题？**  
详情请见：详解js闭包（https://segmentfault.com/a/1190000000652891）  
**js事件处理程序问题？**  
详情请见：JavaScript学习总结（九）事件详解（https://segmentfault.com/a/1190000002174034）  
**eval是做什么的？**  
它的功能是把对应的字符串解析成JS代码并运行；  
应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。  
**JavaScript原型，原型链 ? 有什么特点？**  
\*  原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。  
\*  原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。  
**事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？**  
1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。    
2. 事件处理机制：IE是事件冒泡、firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件。；  
3.  ev.stopPropagation();注意旧ie的方法 ev.cancelBubble = true;  
**ajax 是什么?ajax 的交互模型?同步和异步的区别?如何解决跨域问题?**  
详情请见：JavaScript学习总结（七）Ajax和Http状态字（https://segmentfault.com/a/1190000000691919）  
1. 通过异步模式，提升了用户体验  
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用  
3. Ajax在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。  
4. Ajax的最大的特点是什么。  
  Ajax可以实现动态不刷新（局部刷新）  
  readyState属性 状态 有5个可取值： 0=未初始化 ，1=启动 2=发送，3=接收，4=完成  
ajax的缺点  
  1、ajax不支持浏览器back按钮。  
  2、安全问题 AJAX暴露了与服务器交互的细节。  
  3、对搜索引擎的支持比较弱。  
  4、破坏了程序的异常机制。  
  5、不容易调试。  
跨域： jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面  
js对象的深度克隆  
  function clone(Obj) {     
        var buf;     
        if (Obj instanceof Array) {     
            buf = [];  //创建一个空的数组   
            var i = Obj.length;     
            while (i--) {     
                buf[i] = clone(Obj[i]);     
            }     
            return buf;     
        }else if (Obj instanceof Object){     
            buf = {};  //创建一个空对象   
            for (var k in Obj) {  //为这个对象添加新的属性   
                buf[k] = clone(Obj[k]);     
            }     
            return buf;     
        }else{     
            return Obj;     
        }     
    }    
**AMD和CMD 规范的区别？**  
详情请见：详解JavaScript模块化开发（https://segmentfault.com/a/1190000000733959）  
**网站重构的理解？**  
网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。  
**对于传统的网站来说重构通常是：**  
表格(table)布局改为DIV+CSS  
使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)  
对于移动平台的优化  
针对于SEO进行优化  
深层次的网站重构应该考虑的方面  
减少代码间的耦合  
让代码保持弹性  
严格按规范编写代码  
设计可扩展的API  
代替旧有的框架、语言(如VB)  
增强用户体验  
通常来说对于速度的优化也包含在重构中  
压缩JS、CSS、image等前端资源(通常是由服务器来解决)  
程序的性能优化(如数据读写)  
采用CDN来加速资源加载  
对于JS DOM的优化  
HTTP服务器的文件缓存  
**如何获取UA？**  
<script>   
    function whatBrowser() {    
        document.Browser.Name.value=navigator.appName;    
        document.Browser.Version.value=navigator.appVersion;    
        document.Browser.Code.value=navigator.appCodeName;    
        document.Browser.Agent.value=navigator.userAgent;    
    }    
</script>  
**js数组去重**  
以下是数组去重的三种方法：  
Array.prototype.unique1 = function () {  
  var n = []; //一个新的临时数组  
  for (var i = 0; i < this.length; i++) //遍历当前数组  
  {  
    //如果当前数组的第i已经保存进了临时数组，那么跳过，  
    //否则把当前项push到临时数组里面  
    if (n.indexOf(this[i]) == -1) n.push(this[i]);  
  }  
  return n;  
}  
Array.prototype.unique2 = function()  
{  
    var n = {},r=[]; //n为hash表，r为临时数组  
    for(var i = 0; i < this.length; i++) //遍历当前数组  
    {  
        if (!n[this[i]]) //如果hash表中没有当前项  
        {  
            n[this[i]] = true; //存入hash表  
            r.push(this[i]); //把当前数组的当前项push到临时数组里面  
        }  
    }  
    return r;  
}  
Array.prototype.unique3 = function()  
{  
    var n = [this[0]]; //结果数组  
    for(var i = 1; i < this.length; i++) //从第二项开始遍历  
    {  
        //如果当前数组的第i项在当前数组中第一次出现的位置不是i，  
        //那么表示第i项是重复的，忽略掉。否则存入结果数组  
        if (this.indexOf(this[i]) == i) n.push(this[i]);  
    }  
    return n;  
}  
**HTTP状态码**  
100  Continue  继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息  
200  OK   正常返回信息  
201  Created  请求成功并且服务器创建了新的资源  
202  Accepted  服务器已接受请求，但尚未处理  
301  Moved Permanently  请求的网页已永久移动到新位置。  
302 Found  临时性重定向。  
303 See Other  临时性重定向，且总是使用 GET 请求新的 URI。  
304  Not Modified  自从上次请求后，请求的网页未修改过。  
400 Bad Request  服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。  
401 Unauthorized  请求未授权。  
403 Forbidden  禁止访问。  
404 Not Found  找不到如何与 URI 相匹配的资源。  
500 Internal Server Error  最常见的服务器端错误。  
503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。  
**js操作获取和设置cookie**  
//创建cookie  
function setCookie(name, value, expires, path, domain, secure) {  
    var cookieText = encodeURIComponent(name) + '=' + encodeURIComponent(value);  
    if (expires instanceof Date) {  
        cookieText += '; expires=' + expires;  
    }  
    if (path) {  
        cookieText += '; expires=' + expires;  
    }  
    if (domain) {  
        cookieText += '; domain=' + domain;  
    }  
    if (secure) {  
        cookieText += '; secure';  
    }  
    document.cookie = cookieText;  
}  
//获取cookie  
function getCookie(name) {  
    var cookieName = encodeURIComponent(name) + '=';  
    var cookieStart = document.cookie.indexOf(cookieName);  
    var cookieValue = null;  
    if (cookieStart > -1) {  
        var cookieEnd = document.cookie.indexOf(';', cookieStart);  
        if (cookieEnd == -1) {  
            cookieEnd = document.cookie.length;  
        }  
        cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));  
    }  
    return cookieValue;  
}  
//删除cookie  
function unsetCookie(name) {  
    document.cookie = name + "= ; expires=" + new Date(0);  
}  
**说说TCP传输的三次握手策略**  
为了准确无误地把数据送达目标处，TCP协议采用了三次握手策略。用TCP协议把数据包送出去后，TCP不会对传送  后的情况置之不理，它一定会向对方确认是否成功送达。握手过程中使用了TCP的标志：SYN和ACK。  
发送端首先发送一个带SYN标志的数据包给对方。接收端收到后，回传一个带有SYN/ACK标志的数据包以示传达确认信息。最后，发送端再回传一个带ACK标志的数据包，代表“握手”结束  
若在握手过程中某个阶段莫名中断，TCP协议会再次以相同的顺序发送相同的数据包。  
**说说你对Promise的理解**  
依照 Promise/A+ 的定义，Promise 有四种状态：  
pending: 初始状态, 非 fulfilled 或 rejected.  
fulfilled: 成功的操作.  
rejected: 失败的操作.  
settled: Promise已被fulfilled或rejected，且不是pending  
另外， fulfilled 与 rejected 一起合称 settled。  
Promise 对象用来进行延迟(deferred) 和异步(asynchronous ) 计算。  
Promise 的构造函数  
构造一个 Promise，最基本的用法如下：  
var promise = new Promise(function(resolve, reject) {  
    if (...) {  // succeed  
        resolve(result);  
    } else {   // fails  
        reject(Error(errMessage));  
    }  
});  
Promise 实例拥有 then 方法（具有 then 方法的对象，通常被称为 thenable）。它的使用方法如下：  
promise.then(onFulfilled, onRejected)  
接收两个函数作为参数，一个在 fulfilled 的时候被调用，一个在 rejected 的时候被调用，接收参数就是 future，onFulfilled 对应 resolve, onRejected 对应 reject。  
**Javascript垃圾回收方法**  
标记清除（mark and sweep）  
这是JavaScript最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”。  
垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了  
引用计数(reference counting)  
在低版本IE中经常会出现内存泄露，很多时候就是因为其采用引用计数方式进行垃圾回收。引用计数的策略是跟踪记录每个值被使用的次数，当声明了一个 变量并将一个引用类型赋值给该变量的时候这个值的引用次数就加1，如果该变量的值变成了另外一个，则这个值得引用次数减1，当这个值的引用次数变为0的时 候，说明没有变量在使用，这个值没法被访问了，因此可以将其占用的空间回收，这样垃圾回收器会在运行的时候清理掉引用次数为0的值占用的空间。  
在IE中虽然JavaScript对象通过标记清除的方式进行垃圾回收，但BOM与DOM对象却是通过引用计数回收垃圾的，也就是说只要涉及BOM及DOM就会出现循环引用问题。  
**谈谈性能优化问题**  
代码层面：避免使用css表达式，避免使用高级选择器，通配选择器。 缓存利用：缓存Ajax，使用CDN，使用外部js和css文件以便缓存，添加Expires头，服务端配置Etag，减少DNS查找等 请求数量：合并样式和脚本，使用css图片精灵，初始首屏之外的图片资源按需加载，静态资源延迟加载。 请求带宽：压缩文件，开启GZIP，  
移动端性能优化  
尽量使用css3动画，开启硬件加速。适当使用touch事件代替click事件。避免使用css3渐变阴影效果。 尽可能少的使用box-shadow与gradients。box-shadow与gradients往往都是页面的性能杀手

**什么是Etag？**  
浏览器下载组件的时候，会将它们存储到浏览器缓存中。如果需要再次获取相同的组件，浏览器将检查组件的缓存时间， 假如已经过期，那么浏览器将发送一个条件GET请求到服务器，服务器判断缓存还有效，则发送一个304响应， 告诉浏览器可以重用缓存组件。  
那么服务器是根据什么判断缓存是否还有效呢?答案有两种方式，一种是前面提到的ETag，另一种是根据Last-Modified  
**Expires和Cache-Control**  
Expires要求客户端和服务端的时钟严格同步。HTTP1.1引入Cache-Control来克服Expires头的限制。如果max-age和Expires同时出现，则max-age有更高的优先级。  
Cache-Control: no-cache, private, max-age=0  
ETag: abcde  
Expires: Thu, 15 Apr 2014 20:00:00 GMT  
Pragma: private  
Last-Modified: $now // RFC1123 format  
**栈和队列的区别?**  
栈的插入和删除操作都是在一端进行的，而队列的操作却是在两端进行的。  
队列先进先出，栈先进后出。  
栈只允许在表尾一端进行插入和删除，而队列只允许在表尾一端进行插入，在表头一端进行删除   
**栈和堆的区别？**  
栈区（stack）—   由编译器自动分配释放   ，存放函数的参数值，局部变量的值等。  
堆区（heap）   —   一般由程序员分配释放，   若程序员不释放，程序结束时可能由OS回收。  
堆（数据结构）：堆可以被看成是一棵树，如：堆排序；  
栈（数据结构）：一种先进后出的数据结构。   
**关于Http 2.0 你知道多少？**  
HTTP/2引入了“服务端推（serverpush）”的概念，它允许服务端在客户端需要数据之前就主动地将数据发送到客户端缓存中，从而提高性能。 HTTP/2提供更多的加密支持 HTTP/2使用多路技术，允许多个消息在一个连接上同时交差。 它增加了头压缩（header compression），因此即使非常小的请求，其请求和响应的header都只会占用很小比例的带宽。