## 目录

1. [前言](#user-content-preface)
2. [HTML 部分](#user-content-html)
3. [CSS 部分](#user-content-css)
4. [JavaScript 部分](#user-content-js)
5. [其他问题](#user-content-other)
6. [优质网站推荐](#user-content-web)

## 前言

本文总结了一些优质的前端面试题（多数源于网络），初学者阅后也要用心钻研其中的原理，重要知识需要系统学习，透彻学习，形成自己的知识链。万不可投机取巧，只求面试过关是错误的！

### 面试有几点需注意：(来源程劭非老师 github:@wintercn)

1. 面试题目： 根据你的等级和职位变化，入门级到专家级：范围↑、深度↑、方向↑。
2. 题目类型： 技术视野、项目细节、理论知识题，算法题，开放性题，案例题。
3. 进行追问： 可以确保问到你开始不懂或面试官开始不懂为止，这样可以大大延展题目的区分度和深度，知道你的实际能力。因为这种关联知识是长时期的学习，绝对不是临时记得住的。
4. 回答问题再棒，面试官（可能是你的直接领导面试），会考虑我要不要这个人做我的同事？所以态度很重要。（感觉更像是相亲）
5. 资深的工程师能把absolute和relative弄混，这样的人不要也罢，因为团队需要的你这个人具有可以依靠的才能（靠谱）。

#### 前端开发面试知识点大纲

HTML/CSS：  
对Web标准的理解、浏览器内核差异、兼容性、hack、CSS基本功：布局、盒子模型、  
选择器优先级及使用、HTML5、CSS3、移动端适应  
  
JavaScript：  
数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、  
内存泄漏、事件机制、异步装载回调、模板引擎、Nodejs、JSON、ajax等。  
  
其他：  
HTTP、安全、正则、优化、重构、响应式、移动端、团队协作、可维护、SEO、UED、架构、职业生涯

**备注：**

根据自己需要选择性阅读，面试题是对理论知识的总结，让自己学会应该如何表达。

资料答案不够正确和全面，**欢迎补充**答案、题目；最好是现在网上没有的。

格式不断修改更新中。

## HTML

* Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

（1）<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器的解析器，  
用什么文档类型 规范来解析这个文档。  
  
（2）严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。  
  
（3）在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。  
  
（4）DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

* HTML5 为什么只需要写 '!DOCTYPE HTML'？

HTML5 为什么只需要写 '<!DOCTYPE HTML>'？  
  
HTML5 不基于 SGML，因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；  
  
而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。

* 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

（1）CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，  
比如div默认display属性值为“block”，成为“块级”元素；  
span默认display属性值为“inline”，是“行内”元素。  
  
（2）行内元素有：a b span img input select strong(强调的语气);  
块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p  
  
（3）知名的空元素：  
<br> <hr> <img> <input> <link> <meta>  
鲜为人知的是：  
<area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

* 请问css中的区块 inline inline-block block 三者有什么区别呢？

这样先讲内联元素和块级元素：  
内联元素是不可以控制宽和高、margin等；并且在同一行显示，不换行。  
块级元素时可以控制宽和高、margin等，并且会换行。  
  
inline：使用此属性后，元素会被显示为内联元素，元素则不会换行。  
block：使用此属性后，元素会被现实为块级元素，元素会进行换行。  
inline-block：是使元素以块级元素的形式呈现在行内。  
意思就是说，让这个元素显示在同一行不换行，但是又可以控制高度和宽度，这相当于内联元素的增强。  
  
要注意的是IE6 不支持inline-block

* link 和@import 的区别是？

（1）link属于XHTML标签，而@import是CSS提供的;  
  
（2）页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;  
  
（3）import只在IE5以上才能识别，而link是XHTML标签，无兼容问题;  
  
（4）link方式的样式的权重 高于@import的权重.

* 介绍一下你对浏览器内核的理解？

主要分成两部分：渲染引擎(layout engineer或Rendering Engine)和JS引擎。  
  
渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入CSS等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。  
  
JS引擎则：解析和执行javascript来实现网页的动态效果。  
  
最开始渲染引擎和JS引擎并没有区分的很明确，后来JS引擎越来越独立，内核就倾向于只指渲染引擎。

* 浏览器的内核分别是什么?

Trident内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称MSHTML]  
  
Gecko内核：Netscape6及以上版本，FF,MozillaSuite/SeaMonkey等  
  
Presto内核：Opera7及以上。 [Opera内核原为：Presto，现为：Blink;]  
  
Webkit内核：Safari,Chrome等。 [Chrome的：Blink（WebKit的分支）]  
  
详细文章：[浏览器内核的解析和对比](http://www.cnblogs.com/fullhouse/archive/2011/12/19/2293455.html)

* 常见兼容性问题？

- png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.  
  
- 浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。  
  
- IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。  
  
浮动ie产生的双倍距离 #box{ float:left; width:10px; margin:0 0 0 100px;}  
  
这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中  
加入 ——\_display:inline;将其转化为行内属性。(\_这个符号只有ie6会识别)  
  
渐进识别的方式，从总体中逐渐排除局部。  
  
首先，巧妙的使用“\9”这一标记，将IE游览器从所有情况中分离出来。  
接着，再次使用“+”将IE8和IE7、IE6分离开来，这样IE8已经独立识别。  
  
//css  
.bb{  
 background-color:#f1ee18;/\*所有识别\*/  
 .background-color:#00deff\9; /\*IE6、7、8识别\*/  
 +background-color:#a200ff;/\*IE6、7识别\*/  
 \_background-color:#1e0bd1;/\*IE6识别\*/  
}  
  
- IE下,可以使用获取常规属性的方法来获取自定义属性,  
也可以使用getAttribute()获取自定义属性;  
Firefox下,只能使用getAttribute()获取自定义属性.  
解决方法:统一通过getAttribute()获取自定义属性.  
  
- IE下,even对象有x,y属性,但是没有pageX,pageY属性;  
Firefox下,event对象有pageX,pageY属性,但是没有x,y属性.  
  
- 解决方法：（条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。  
  
- Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,  
可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决.  
  
超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了解决方法是改变CSS属性的排列顺序:  
L-V-H-A : a:link {} a:visited {} a:hover {} a:active {}

* html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？

- HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。  
  
- 绘画 canvas  
用于媒介回放的 video 和 audio 元素  
本地离线存储 localStorage (目前业界基本上统一为5M)长期存储数据，浏览器关闭后数据不丢失；  
sessionStorage 的数据在浏览器关闭后自动删除  
  
语意化更好的内容元素，比如 article、footer、header、nav、section  
表单控件，calendar、date、time、email、url、search  
新的技术webworker, websockt, Geolocation  
  
- 移除的元素  
  
纯表现的元素：basefont，big，center，font, s，strike，tt，u；  
  
对可用性产生负面影响的元素：frame，frameset，noframes；  
  
支持HTML5新标签：  
  
- IE8/IE7/IE6支持通过document.createElement方法产生的标签，  
 可以利用这一特性让这些浏览器支持HTML5新标签，  
  
浏览器支持新标签后，还需要添加标签默认的样式：  
  
- 当然最好的方式是直接使用成熟的框架、使用最多的是html5shim框架  
 <!--[if lt IE 9]>  
 <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>  
 <![endif]-->  
如何区分： DOCTYPE声明\新增的结构元素\功能元素

* 语义化的理解？

用正确的标签做正确的事情！  
html语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；  
在没有样式CCS情况下也以一种文档格式显示，并且是容易阅读的。  
搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO。  
使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。

* HTML5的离线储存怎么使用，工作原理能不能解释一下？

在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。  
  
原理：HTML5的离线存储是基于一个新建的.appcache文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像cookie一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。  
  
如何使用：  
1、页面头部像下面一样加入一个manifest的属性；  
2、在cache.manifest文件的编写离线存储的资源；  
 CACHE MANIFEST  
 #v0.11  
 CACHE:  
 js/app.js  
 css/style.css  
 NETWORK:  
 resourse/logo.png  
 FALLBACK:  
 / /offline.html  
3、在离线状态时，操作window.applicationCache进行需求实现。  
  
详细的使用请参考：[有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617)  
  
localStorage 长期存储数据，浏览器关闭后数据不丢失；  
sessionStorage 数据在浏览器关闭后自动删除。

* 浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢？

在线的情况下，浏览器发现html头部有manifest属性，它会请求manifest文件，如果是第一次访问app，那么浏览器就会根据manifest文件的内容下载相应的资源并且进行离线存储。如果已经访问过app并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的manifest文件与旧的manifest文件，如果文件没有发生改变，就不做任何操作，如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。  
  
离线的情况下，浏览器就直接使用离线存储的资源。  
  
详细的使用请参考：[有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617)

* (写)描述一段语义的html代码吧。

（HTML5中新增加的很多标签（如：<article>、<nav>、<header>和<footer>等）  
就是基于语义化设计原则）  
< div id="header">  
< h1>标题< /h1>  
< h2>专注Web前端技术< /h2>  
< /div>

* iframe有那些缺点？

\*iframe会阻塞主页面的Onload事件；  
\*搜索引擎的检索程序无法解读这种页面，不利于SEO;  
  
\*iframe和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。  
  
使用iframe之前需要考虑这两个缺点。如果需要使用iframe，最好是通过javascript  
动态给iframe添加src属性值，这样可以绕开以上两个问题。

* 请描述一下 cookies，sessionStorage 和 localStorage 的区别？

cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。  
  
cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。  
  
sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。  
  
存储大小：  
 cookie数据大小不能超过4k。  
 sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。  
  
有期时间：  
 localStorage 存储持久数据(目前业界基本上统一为5M)，浏览器关闭后数据不丢失除非主动删除数据；  
 sessionStorage 数据在当前浏览器窗口关闭后自动删除。  
 cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭  
  
cookie在浏览器和服务器间来回传递。 sessionStorage和localStorage不会  
sessionStorage和localStorage的存储空间更大；  
sessionStorage和localStorage有更多丰富易用的接口；  
sessionStorage和localStorage各自独立的存储空间；

* Label的作用是什么？是怎么用的？

label标签来定义表单控制间的关系,当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。  
  
<label for="Name">Number:</label>  
<input type="text" name="Name" id="Name"/>  
  
<label>Date:<input type="text" name="B"/></label>

* HTML5的form如何关闭自动完成功能？

给不想要提示的 form 或下某个input 设置为 autocomplete=off。

* 如何实现浏览器内多个标签页之间的通信? (阿里)

使用localStorage、使用cookie+setInterval

* webSocket如何兼容低浏览器？(阿里)

Adobe Flash Socket 、  
ActiveX HTMLFile (IE) 、  
基于 multipart 编码发送 XHR 、  
基于长轮询的 XHR

* 页面可见性（Page Visibility）API 可以有哪些用途？

在页面被切换到其他后台进程的时候，自动暂停音乐或视频的播放

* 如何在页面上实现一个圆形的可点击区域？

1、map+area或者svg  
2、border-radius  
3、纯js实现 需要求一个点在不在圆上简单算法、获取鼠标坐标等等

* 实现不使用 border,画出1px高的线，在不同浏览器的标准模式与怪异模式下都能保持一致的效果

<div style="height:1px;overflow:hidden;background:#ccc"></div>

* 网页验证码是干嘛的，是为了解决什么安全问题。

区分用户是计算机还是人的公共全自动程序。可以防止：恶意破解密码、刷票、论坛灌水；  
有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试；

## CSS

* 介绍一下CSS的盒子模型？

（1）有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 pading;  
  
（2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border).

* CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些？

选择符:  
1.id选择器（ # myid）  
2.类选择器（.myclassname）  
3.标签选择器（div, h1, p）  
4.相邻选择器（h1 + p）  
5.子选择器（ul < li）  
6.后代选择器（li a）  
7.通配符选择器（ \* ）  
8.属性选择器（a[rel = "external"]）  
9.伪类选择器（a: hover, li: nth - child）  
  
可继承的样式：  
font-size font-family color, UL LI DL DD DT;  
  
不可继承的样式：  
border padding margin width height ;  
  
优先级就近原则，同权重情况下样式定义最近者为准;  
  
载入样式以最后载入的定位为准;  
  
优先级为:  
  
 !important > id > class > tag  
  
 important 比 内联优先级高  
  
CSS3新增伪类举例：  
  
p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。  
p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。  
p:only-of-type选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。  
p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。  
p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。  
:enabled :disabled 控制表单控件的禁用状态。  
:checked 单选框或复选框被选中。

* 如何居中div？如何居中一个浮动元素？

\* 给div设置一个宽度，然后添加margin:0 auto属性  
  
div{  
 width:200px;  
 margin:0 auto;  
}  
  
\* 居中一个浮动元素  
  
 确定容器的宽高 宽500 高 300 的层  
 设置层的外边距  
  
.div {  
 Width:500px ; height:300px;//高度可以不设  
 Margin: -150px 0 0 -250px;  
 position:relative;//相对定位  
 background-color:pink;//方便看效果  
 left:50%;  
 top:50%;  
}  
  
\* 让绝对定位的div居中  
  
 position: absolute;  
 width: 1200px;  
 background: none;  
 margin: 0 auto;  
 top: 0;  
 left: 0;  
 bottom: 0;  
 right: 0;

* 列出display的值，说明他们的作用。

block 象块类型元素一样显示。  
 none 缺省值。象行内元素类型一样显示。  
 inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。  
 list-item 象块类型元素一样显示，并添加样式列表标记。

* position的值， relative和absolute定位原点是？

absolute  
生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。  
  
 fixed （老IE不支持）  
生成绝对定位的元素，相对于浏览器窗口进行定位。  
  
 relative  
生成相对定位的元素，相对于其正常位置进行定位。  
  
 static  
默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right z-index 声明）。  
  
 sticky  
粘性定位，这是一个结合了 position:relative 和 position:fixed 两种定位功能于一体的特殊定位，适用于一些特殊场景。元素先按照普通文档流定位，然后相对于该元素在流中的 flow root（BFC）和 containing block（最近的块级祖先元素）定位。而后，元素定位表现为在跨越特定阈值前为相对定位，之后为固定定位。  
  
 inherit  
规定从父元素继承 position 属性的值。

* CSS3有哪些新特性？

CSS3实现  
圆角（border-radius:8px）,  
阴影（box-shadow:10px）,  
对文字加特效（text-shadow）,  
线性渐变（gradient）,  
旋转（transform）  
transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,倾斜  
增加了更多的CSS选择器 多背景 rgba

* 请解释一下CSS3的Flexbox（弹性盒布局模型）,以及适用场景？ ``` 文章链接： http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html?utm\_source=tuicool（语法篇） http://www.ruanyifeng.com/blog/2015/07/flex-examples.html（实例篇）

Flex是Flexible Box的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。 布局的传统解决方案，基于盒状模型，依赖 display属性 + position属性 + float属性。它对于那些特殊布局非常不方便，比如，垂直居中就不容易实现。 简单的分为容器属性和元素属性 容器的属性： flex-direction：决定主轴的方向（即子item的排列方法） .box { flex-direction: row | row-reverse | column | column-reverse; }

flex-wrap：决定换行规则 .box{ flex-wrap: nowrap | wrap | wrap-reverse; }

flex-flow：让弹性盒的元素以相反的顺序显示，且在必要的时候进行拆行： .box { flex-flow: flex-direction flex-wrap|initial|inherit;; }

justify-content：对其方式，水平主轴对齐方式 align-items：对齐方式，竖直轴线方向

项目的属性（元素的属性）：

order属性：定义项目的排列顺序，顺序越小，排列越靠前，默认为0 flex-grow属性：定义项目的放大比例，即使存在空间，也不会放大 flex-shrink属性：定义了项目的缩小比例，当空间不足的情况下会等比例的缩小，如果定义个item的flow-shrink为0，则为不缩小 flex-basis属性：定义了在分配多余的空间，项目占据的空间。 flex：是flex-grow和flex-shrink、flex-basis的简写，默认值为0 1 auto。 align-self：允许单个项目与其他项目不一样的对齐方式，可以覆盖align-items，默认属性为auto，表示继承父元素的align-items

比如说，用flex实现圣杯布局 ```

* 用纯CSS创建一个三角形的原理是什么？

把上、左、右三条边隐藏掉（颜色设为 transparent）  
#demo {  
 width: 0;  
 height: 0;  
 border-width: 20px;  
 border-style: solid;  
 border-color: transparent transparent red transparent;  
}

* 一个满屏 品 字布局 如何设计?

简单的方式：  
上面的div宽100%，  
下面的两个div分别宽50%，  
然后用float或者inline使其不换行即可

* 垂直居中的方法

1. margin:auto法 ``` div{ width: 400px; height: 400px; position: relative; border: 1px solid #465468; } img{ position: absolute; margin: auto; top: 0; left: 0; right: 0; bottom: 0; }

定位为上下左右为0，margin：0可以实现脱离文档流的居中.

(2) margin负值法

.container{ width: 500px; height: 400px; border: 2px solid #379; position: relative; } .inner{ width: 480px; height: 380px; background-color: #746; position: absolute; top: 50%; left: 50%; margin-top: -190px; /*height的一半*/ margin-left: -240px; /*width的一半*/ }

(3)利用flex  
  
将父元素设置为display:flex，并且设置align-items:center;justify-content:center;

.container{ width: 300px; height: 200px; border: 3px solid #546461; display: -webkit-flex; display: flex; -webkit-align-items: center; align-items: center; -webkit-justify-content: center; justify-content: center; } .inner{ border: 3px solid #458761; padding: 20px; } ```

* 为什么要初始化CSS样式。

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。  
  
当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。  
  
最简单的初始化方法就是： \* {padding: 0; margin: 0;} （不建议）  
  
淘宝的样式初始化：  
body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt,  
dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }  
body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }  
h1, h2, h3, h4, h5, h6{ font-size:100%; }  
address, cite, dfn, em, var { font-style:normal; }  
code, kbd, pre, samp { font-family:couriernew, courier, monospace; }  
small{ font-size:12px; }  
ul, ol { list-style:none; }  
a { text-decoration:none; }  
a:hover { text-decoration:underline; }  
sup { vertical-align:text-top; }  
sub{ vertical-align:text-bottom; }  
legend { color:#000; }  
fieldset, img { border:0; }  
button, input, select, textarea { font-size:100%; }  
table { border-collapse:collapse; border-spacing:0; }

* li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？
* 经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧 ？
* absolute的containing block计算方式跟正常流有什么不同？

无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：  
1、若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；  
2、否则,则由这个祖先元素的 padding box 构成。  
如果都找不到，则为 initial containing block。  
  
补充：  
1. static(默认的)/relative：简单说就是它的父元素的内容框（即去掉padding的部分）  
2. absolute: 向上找最近的定位为absolute/relative的元素  
3. fixed: 它的containing block一律为根元素(html/body)，根元素也是initial containing block

* position跟display、margin collapse、overflow、float这些特性相互叠加后会怎么样？
* 对BFC规范(块级格式化上下文：block formatting context)的理解？

W3C CSS 2.1 规范中的一个概念,它是一个独立容器，决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用。）  
 一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。  
 不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,因此Box内的元素会以不同的方式渲染,也就是说BFC内部的元素和外部的元素不会互相影响。

* css定义的权重

以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：  
  
/\*权重为1\*/  
div{  
}  
/\*权重为10\*/  
.class1{  
}  
/\*权重为100\*/  
#id1{  
}  
/\*权重为100+1=101\*/  
#id1 div{  
}  
/\*权重为10+1=11\*/  
.class1 div{  
}  
/\*权重为10+10+1=21\*/  
.class1 .class2 div{  
}  
  
如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现

* 解释下浮动和它的工作原理？清除浮动的技巧
* 用过媒体查询，针对移动端的布局吗？
* 使用 CSS 预处理器吗？喜欢那个？

SASS (SASS、LESS没有本质区别，只因为团队前端都是用的SASS)

* CSS优化、提高性能的方法有哪些？
* 浏览器是怎样解析CSS选择器的？
* 在网页中的应该使用奇数还是偶数的字体？为什么呢？
* margin和padding分别适合什么场景使用？
* 抽离样式模块怎么写，说出思路，有无实践经验？[阿里航旅的面试题]
* 元素竖向的百分比设定是相对于容器的高度吗？
* 全屏滚动的原理是什么？用到了CSS的那些属性？
* 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？
* 视差滚动效果，如何给每页做不同的动画？（回到顶部，向下滑动要再次出现，和只出现一次分别怎么做？）
* ::before 和 :after中双冒号和单冒号 有什么区别？解释一下这2个伪元素的作用。
* 如何修改chrome记住密码后自动填充表单的黄色背景 ？
* 你对line-height是如何理解的？
* 设置元素浮动后，该元素的display值是多少？（自动变成display:block）
* 怎么让Chrome支持小于12px 的文字？
* 让页面里的字体变清晰，变细用CSS怎么做？（-webkit-font-smoothing: antialiased;）
* font-style属性可以让它赋值为“oblique” oblique是什么意思？
* position:fixed;在android下无效怎么处理？
* 如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）

多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60＊1000ms ＝ 16.7ms

* display:inline-block 什么时候会显示间隙？(携程)

移除空格、使用margin负值、使用font-size:0、letter-spacing、word-spacing

* overflow: scroll时不能平滑滚动的问题怎么处理？
* 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度。
* png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？
* 什么是Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）

如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，  
所以不如隔离开。  
  
因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，  
这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。  
  
同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，  
提高了webserver的http请求的解析速度。

## JavaScript

* 介绍js的数据类型。

ECMAScript中有5中简单数据类型（也称为基本数据类型）: Undefined、Null、Boolean、Number和String。  
还有1中复杂的数据类型————Object，Object本质上是由一组无序的名值对组成的。  
  
其中Undefined、Null、Boolean、Number都属于基本类型。  
  
Object、Array和Function则属于引用类型，  
  
String有些特殊（String类型用于表示由零或多个16位Unicode字符组成的字符序列，即字符串）（string类型有些特殊，因为字符串具有可变的大小，所以显然它不能被直接存储在具有固定大小的变量中。由于效率的原因，我们希望JS只复制对字符串的引用，而不是字符串的内容。但是另一方面，字符串在许多方面都和基本类型的表现相似，而字符串是不可变的这一事实（即没法改变一个字符串值的内容），因此可以将字符串看成行为与基本类型相似的不可变引用类型）  
  
拓展：原始数据类型Symbol 表示独一无二的值。

* 介绍js有哪些内置对象？

Object 是 JavaScript 中所有对象的父对象  
  
数据封装类对象：Object、Array、Boolean、Number 和 String  
其他对象：Function、Arguments、Math、Date、RegExp、Error

* 说几条写JavaScript的基本规范？

1.不要在同一行声明多个变量。  
2.请使用 ===/!==来比较true/false或者数值  
3.使用对象字面量替代new Array这种形式  
4.不要使用全局函数。  
5.Switch语句必须带有default分支  
6.函数不应该有时候有返回值，有时候没有返回值。  
7.For循环必须使用大括号  
8.If语句必须使用大括号  
9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。

* JavaScript原型，原型链 ? 有什么特点？

每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，  
如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，  
于是就这样一直找下去，也就是我们平时所说的原型链的概念。  
关系：instance.constructor.prototype = instance.\_\_proto\_\_  
  
特点：  
JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。  
  
当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。  
  
 function Func(){}  
  
 Func.prototype.name = "Sean";  
 Func.prototype.getInfo = function() {  
 return this.name;  
 }  
 var person = new Func();//现在可以参考var person = Object.create(oldObject);  
  
 console.log(person.getInfo());//它拥有了Func的属性和方法  
 //"Sean"  
  
 console.log(Func.prototype);  
 // Func { name="Sean", getInfo=function()}  
  
 //再看下这个  
 function A(){}  
 var a = new A();  
 console.log(a.\_\_proto\_\_);  
 console.log(a.\_\_proto\_\_.\_\_proto\_\_);  
 console.log(a.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_);

* JavaScript有几种类型的值？，你能画一下他们的内存图吗？

栈：原始数据类型（Undefined，Null，Boolean，Number、String）  
堆：引用数据类型（对象、数组和函数）  
  
两种类型的区别是：存储位置不同；  
原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；  
引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其  
在栈中的地址，取得地址后从堆中获得实体  
  
![Stated Clearly Image](http://www.w3school.com.cn/i/ct\_js\_value.gif)

* Javascript如何实现继承？

1、构造继承  
2、原型继承  
3、实例继承  
4、拷贝继承  
  
原型prototype机制或apply和call方法去实现较简单，  
  
建议使用构造函数与原型混合方式。  
  
//类式继承  
function Super(){  
 this.color = ['red','blue'];  
}  
  
function Sub(){  
 Super.call(this);  
}  
  
//原型继承  
function Parent(){  
 this.name = 'wang';  
}  
  
function Child(){  
 this.age = 28;  
}  
Child.prototype = new Parent();//继承了Parent，通过原型  
  
var demo = new Child();  
alert(demo.age);  
alert(demo.name);//得到被继承的属性

* javascript创建对象的几种方式？

javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。  
  
  
1、对象字面量的方式  
  
 person={  
 firstname:"Mark",  
 lastname:"Yun",  
 age:25,  
 eyecolor:"black"  
 };  
  
2、用function来模拟无参的构造函数  
  
 function Person(){}  
 var person = new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class  
 person.name="Mark";  
 person.age="25";  
 person.work=function(){  
 alert(person.name+" hello...");  
 }  
 person.work();  
  
3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）  
  
 function Pet(name,age,hobby){  
 this.name=name;//this作用域：当前对象  
 this.age=age;  
 this.hobby=hobby;  
 this.eat=function(){  
 alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");  
 }  
 }  
 var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象  
 maidou.eat();//调用eat方法  
  
  
4、用工厂方式来创建（内置对象）  
  
 var wcDog =new Object();  
 wcDog.name="旺财";  
 wcDog.age=3;  
 wcDog.work=function(){  
 alert("我是"+wcDog.name+",汪汪汪......");  
 }  
 wcDog.work();  
  
  
5、用原型方式来创建  
  
 function Dog(){}  
 Dog.prototype.name="旺财";  
 Dog.prototype.eat=function(){  
 alert(this.name+"是个吃货");  
 }  
 var wangcai =new Dog();  
 wangcai.eat();  
  
  
6、用混合方式来创建  
  
 function Car(name,price){  
 this.name=name;  
 this.price=price;  
 }  
 Car.prototype.sell=function(){  
 alert("我是"+this.name+"，我现在卖"+this.price+"万元");  
 }  
 var camry = new Car("凯美瑞",27);  
 camry.sell();

* Javascript作用链域?

全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。  
当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，  
直至全局函数，这种组织形式就是作用域链。

* 谈谈This对象的理解。
* 什么是window对象? 什么是document对象?
* 看看下面运行结果？

window.onload=function(){  
 var a=1+"1";//11  
 var b="1"+1;//11  
 var c="abc"+12+5+"def";//abc125def  
 var d="abc"+(12+5)+"def";//abc17def  
}

* 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）。

* null，undefined 的区别？

null表示一个对象被定义了，值为“空值”；  
undefined 表示不存在这个值。  
  
typeof undefined  
 //"undefined"  
 undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；  
 例如变量被声明了，但没有赋值时，就等于undefined  
  
typeof null  
 //"object"  
 null : 是一个对象(空对象, 没有任何属性和方法)；  
 例如作为函数的参数，表示该函数的参数不是对象；  
  
注意：  
 在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined  
  
  
再来一个例子：  
  
 null  
 Q：有张三这个人么？  
 A：有！  
 Q：张三有房子么？  
 A：没有！  
  
 undefined  
 Q：有张三这个人么？  
 A：没有！  
  
参考阅读：[undefined与null的区别](http://www.ruanyifeng.com/blog/2014/03/undefined-vs-null.html)  
  
前者的意思是javascript解释器不知道这是什麽东西,会抛出"未定义"的错误  
後者的意思是你定义了它,但它没有分配内存空间,无心肝不痛,不会抛出错误.

* 写一个通用的事件侦听器函数。

// event(事件)工具集，来源：github.com/markyun  
markyun.Event = {  
 // 页面加载完成后  
 readyEvent : function(fn) {  
 if (fn==null) {  
 fn=document;  
 }  
 var oldonload = window.onload;  
 if (typeof window.onload != 'function') {  
 window.onload = fn;  
 } else {  
 window.onload = function() {  
 oldonload();  
 fn();  
 };  
 }  
 },  
 // 视能力分别使用dom0||dom2||IE方式 来绑定事件  
 // 参数： 操作的元素,事件名称 ,事件处理程序  
 addEvent : function(element, type, handler) {  
 if (element.addEventListener) {  
 //事件类型、需要执行的函数、是否捕捉  
 element.addEventListener(type, handler, false);  
 } else if (element.attachEvent) {  
 element.attachEvent('on' + type, function() {  
 handler.call(element);  
 });  
 } else {  
 element['on' + type] = handler;  
 }  
 },  
 // 移除事件  
 removeEvent : function(element, type, handler) {  
 if (element.removeEnentListener) {  
 element.removeEnentListener(type, handler, false);  
 } else if (element.datachEvent) {  
 element.detachEvent('on' + type, handler);  
 } else {  
 element['on' + type] = null;  
 }  
 },  
 // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)  
 stopPropagation : function(ev) {  
 if (ev.stopPropagation) {  
 ev.stopPropagation();  
 } else {  
 ev.cancelBubble = true;  
 }  
 },  
 // 取消事件的默认行为  
 preventDefault : function(event) {  
 if (event.preventDefault) {  
 event.preventDefault();  
 } else {  
 event.returnValue = false;  
 }  
 },  
 // 获取事件目标  
 getTarget : function(event) {  
 return event.target || event.srcElement;  
 },  
 // 获取event对象的引用，取到事件的所有信息，确保随时能使用event；  
 getEvent : function(e) {  
 var ev = e || window.event;  
 if (!ev) {  
 var c = this.getEvent.caller;  
 while (c) {  
 ev = c.arguments[0];  
 if (ev && Event == ev.constructor) {  
 break;  
 }  
 c = c.caller;  
 }  
 }  
 return ev;  
 }  
};

* ["1", "2", "3"].map(parseInt) 答案是多少？

[1, NaN, NaN] 因为 parseInt 需要两个参数 (val, radix) 但 map 传了 3 个 (element, index, array)  
  
 由于map()接收的回调函数可以有3个参数：callback(currentValue, index, array)，通常我们仅需要第一个参数，而忽略了传入的后面两个参数。不幸的是，parseInt(string, radix)没有忽略第二个参数，导致实际执行的函数分别是：  
  
parseInt('0', 0); // 0, 按十进制转换  
  
parseInt('1', 1); // NaN, 没有一进制  
  
parseInt('2', 2); // NaN, 按二进制转换不允许出现2  
  
可以改为["1", "2", "3"].map(Number);，因为Number(value)函数仅接收一个参数。

* 如何创建一个对象? （画出此对象的内存图）

function Person(name, age) {  
 this.name = name;  
 this.age = age;  
 this.sing = function() { alert(this.name) }  
}

* 谈谈This对象的理解。

this是js的一个关键字，随着函数使用场合不同，this的值会发生变化。  
  
但是有一个总原则，那就是this指的是调用函数的那个对象。  
  
this一般情况下：是全局对象Global。 作为方法调用，那么this就是指这个对象

* 事件是？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();（旧ie的方法 ev.cancelBubble = true;）

* 什么是闭包（closure），为什么要用它？

一句话可以概括：闭包就是能够读取其他函数内部变量的函数，或者子函数在外调用，子函数所在的父函数的作用域不会被释放。  
  
闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。  
  
闭包的特性：  
  
1.函数内再嵌套函数  
2.内部函数可以引用外层的参数和变量  
3.参数和变量不会被垃圾回收机制回收  
  
//li节点的onclick事件都能正确的弹出当前被点击的li索引  
 <ul id="testUL">  
 <li> index = 0</li>  
 <li> index = 1</li>  
 <li> index = 2</li>  
 <li> index = 3</li>  
</ul>  
<script type="text/javascript">  
 var nodes = document.getElementsByTagName("li");  
 for(i = 0;i<nodes.length;i+= 1){  
 nodes[i].onclick = function(){  
 console.log(i+1);//不用闭包的话，值每次都是4  
 }(i);  
 }  
</script>

* "use strict";是什么意思 ? 使用它的好处和坏处分别是什么？

use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,  
  
使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。  
默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;  
全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；  
消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;  
  
提高编译器效率，增加运行速度；  
为未来新版本的Javascript标准化做铺垫。

* 如何判断一个对象是否属于某个类？

使用instanceof （待完善）  
  
if(a instanceof Person){  
 alert('yes');  
}

* 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);

* Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

hasOwnProperty  
  
javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。  
使用方法：  
object.hasOwnProperty(proName)  
其中参数object是必选项。一个对象的实例。  
proName是必选项。一个属性名称的字符串值。  
  
如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。

* Node.js的适用场景？

高并发、聊天、实时消息推送

* JSON 的了解？

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。  
它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小  
如：{"age":"12", "name":"back"}

* js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

* Ajax 是什么? 如何创建一个Ajax？

ajax的全称：Asynchronous Javascript And XML。  
异步传输+js+xml。  
所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。  
  
(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

* GET POST 区别

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，  
值和表单内各个字段一一对应，在URL中可以看到。  
post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML  
HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. 对于get方式，服务器端用Request.QueryString获取变量的值，  
对于post方式，服务器端用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。  
但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。  
  
建议：  
1、get方式的安全性较Post方式要差些，包含机密信息的话，建议用Post数据提交方式；  
2、在做数据查询时，建议用Get方式；而在做数据添加、修改或删除时，建议用Post方式；

* 如何解决跨域问题?

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面  
(1) iframe (2) 动态创建script标签 （3）JSONP （4）CORS  
  
[了解更多](http://www.liaoxuefeng.com/wiki/001434446689867b27157e896e74d51a89c25cc8b43bdb3000/001434499861493e7c35be5e0864769a2c06afb4754acc6000)

* 页面编码和被请求的资源编码如果不一致如何处理？
* 模块化怎么做？

[Javascript模块化编程](http://www.ruanyifeng.com/blog/2012/10/javascript_module.html)

//不暴露私有成员  
var module1 = (function(){  
　　var \_count = 0;  
　　var m1 = function(){  
　　　　//...  
　　};  
　　var m2 = function(){  
　　　　//...  
　　};  
　　return {  
　　　　m1 : m1,  
　　　　m2 : m2  
　　};  
})();

* AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

> AMD 规范在这里：https://github.com/amdjs/amdjs-api/wiki/AMD  
  
> CMD 规范在这里：https://github.com/seajs/seajs/issues/242  
  
Asynchronous Module Definition，异步模块定义，所有的模块将被异步加载，模块加载不影响后面语句运行。所有依赖某些模块的语句均放置在回调函数中。  
  
 区别：  
  
 1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.  
 2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：  
  
// CMD  
define(function(require, exports, module) {  
 var a = require('./a')  
 a.doSomething()  
 // 此处略去 100 行  
 var b = require('./b') // 依赖可以就近书写  
 b.doSomething()  
 // ...  
})  
  
// AMD 默认推荐  
define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好  
 a.doSomething()  
 // 此处略去 100 行  
 b.doSomething()  
 // ...  
})

* 异步加载的方式有哪些？

(1) defer，只支持IE  
  
(2) async：  
  
(3) 创建script，插入到DOM中，加载完毕后callBack

* requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何缓存的？）
* 谈一谈你对ECMAScript6的了解？

https://segmentfault.com/a/1190000002583196

http://es6.ruanyifeng.com/

* ECMAScript6 怎么写class，为什么会出现class这种东西?
* 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？

//call: 调用一个对象的一个方法，以另一个对象替换当前对象  
//apply: 应用某一对象的一个方法，用另一个对象替换当前对象  
function add(a,b){  
 return a+b;  
}  
function sub(a,b){  
 return a-b;  
}  
alert(add.call(sub,3,1));//add替换sub  
//call函数和apply方法的第一个参数都是要传入给当前对象的对象，及函数内部的this。后面的参数都是传递给当前对象的参数。  
var func = new function(){  
 this.a = "func";  
}  
var myfunc = function(x){  
 var a = "myfunc";  
 alert(this.a);  
 alert(x);  
}  
myfunc.call(func,"var");  
/\*对于apply和call两者在作用上是相同的，但两者在参数上有区别的。  
对于第一个参数意义都一样，但对第二个参数：  
apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。  
如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3])  
同时使用apply的好处是可以直接将当前函数的arguments对象作为apply的第二个参数传入\*/

* 数组和对象有哪些原生方法，列举一下？
* JS 怎么实现一个类。怎么实例化这个类
* JavaScript中的作用域与变量声明提升？
* 如何编写高性能的Javascript？
* 那些操作会造成内存泄漏？
* JQuery的源码看过吗？能不能简单概况一下它的实现原理？
* jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？
* jquery中如何将数组转化为json字符串，然后再转化回来？

$.fn.stringifyArray = function(array) {  
 return JSON.stringify(array)  
}  
  
$.fn.parseArray = function(array) {  
 return JSON.parse(array)  
}  
//然后调用：  
$("#xxx").stringifyArray(array)

* jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？
* jquery.extend 与 jquery.fn.extend的区别？
* jQuery 的队列是如何实现的？队列可以用在哪些地方？
* 谈一下Jquery中的bind(),live(),delegate(),on()的区别？
* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* 是否知道自定义事件。jQuery里的fire函数是什么意思，什么时候用？
* jQuery 是通过哪个方法和 Sizzle 选择器结合的？（jQuery.fn.find()进入Sizzle）
* 针对 jQuery性能的优化方法？
* jQuery与jQuery UI 有啥区别？

\*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。  
  
\*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。  
提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

* 针对 jQuery 的优化方法？

\*基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。  
  
\*频繁操作的DOM，先缓存起来再操作。用jQuery的链式调用更好。比如：var str=$("a").attr("href");  
  
\*for (var i = size; i < arr.length; i++) {}  
for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：  
for (var i = size, length = arr.length; i < length; i++) {}

* Zepto的点透问题如何解决？
* jQueryUI如何自定义组件?
* 需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退时正确响应。给出你的技术实现方案？
* 如何判断当前脚本运行在浏览器还是node环境中？（阿里）

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

* 移动端最小触控区域是多大？
* jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?
* 把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？
* 移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）
* 知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?
* Underscore 对哪些 JS 原生对象进行了扩展以及提供了哪些好用的函数方法？
* JavaScript中的作用域与变量声明提升？
* 如何编写高性能的Javascript？

https://segmentfault.com/a/1190000011735788

执行与加载、变量处理、DOM操作的注意点、算法和流程控制、

* 那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）

* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* Node.js的适用场景？
* (如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?
* 解释一下 Backbone 的 MVC 实现方式？ 2010年，Backbone.js; Backbone 将前端代码分成两个基本部分。 Model：管理数据、View：数据的展现; Backbone 只有 M 和 V，没有 C。因为，前端 Controller 与后端不同。 不需要，也不应该处理业务逻辑; 只需要处理 UI 逻辑，响应用户的一举一动; 所以，前端 Controller 相对比较简单。Backbone 没有 C，只用事件来处理 UI 逻辑。
* MVVM，MVC 详解 [MVVM](https://github.com/ruanyf/jstraining/blob/master/docs/history.md#mvvm-%E6%A8%A1%E5%BC%8F)

[MVC](https://github.com/ruanyf/jstraining/blob/master/docs/history.md#%E5%89%8D%E7%AB%AF-mvc-%E6%A1%86%E6%9E%B6)

* 什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点? http://xxx#xxx

spa

* 浏览器渲染原理 / 浏览器渲染页面的过程

1. 用户输入URL地址 2. 浏览器解析URL解析出主机名 3. 浏览器将主机名转换成服务器ip地址（浏览器先查找本地DNS缓存列表 没有的话 再向浏览器默认的DNS服务器发送查询请求 同时缓存） 4. 浏览器将端口号从URL中解析出来 5. 浏览器建立一条与目标Web服务器的TCP连接（三次握手） 6. 浏览器向服务器发送一条HTTP请求报文 7. 服务器向浏览器返回一条HTTP响应报文 8. 关闭连接 浏览器解析文档

* 爬虫引擎是怎样抓取页面的
* 异步编程的四种方法 http://www.ruanyifeng.com/blog/2012/12/asynchronous%EF%BC%BFjavascript.html

1、回调函数 2、事件监听 3、发布/订阅 4、Promises对象 - 解释同步、阻塞、并行

同步，异步 同步：在发出一个同步调用时，在没有得到结果之前，该调用就不返回。 同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。 异步：在发出一个异步调用后，调用者不会立刻得到结果，该调用就返回了。 异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

(阻塞)[https://img-blog.csdn.net/20161223093414586?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvc2luYXRfMzU1MTIyNDU=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast] 一个线程/进程经历的5个状态，创建，就绪，运行，阻塞，终止。各个状态的转换条件如上图，其中有个阻塞状态，就是说当线程中调用某个函数，需要IO请求，或者暂时得不到竞争资源的，操作系统会把该线程阻塞起来，避免浪费CPU资源，等到得到了资源，再变成就绪状态，等待CPU调度运行。

阻塞调用是指调用结果返回之前，调用者会进入阻塞状态等待。只有在得到结果之后才会返回。 非阻塞调用是指在不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。

并发，并行 并发是指一个时间段内，有几个程序都在同一个CPU上运行，但任意一个时刻点上只有一个程序在处理机上运行。 并行是指一个时间段内，有几个程序都在几个CPU上运行，任意一个时刻点上，有多个程序在同时运行，并且多道程序之间互不干扰。

* js实现数值千分位

let n = 111111; n.toLocaleString();

* 多语言网站建设应注意哪些事项？
* React非父子、兄弟组件传值
* "123456789876543212345678987654321..."的第n位是什么？
* 浏览器打开一个页面前端缓存了哪些东西？
* 说一下vue的生命周期 https://www.cnblogs.com/penghuwan/p/7192203.html
* 你的博客用的是Koa，Express用过吗？
* 你的博客用的什么服务器？
* 有没有自己实现过Promise？
* 有没有写过Webpack插件？
* new 一个对象后发生了什么？
* 说一下原型和原型链
* 有没有自己写过比较复杂的正则？
* 有没有封装过axios？
* 前后分离的系统，一个请求出错了，如何中断其它请求？
* 如何在axios中添加登陆验证？
* rollup了解过没？ 为什么rollup打包赘余代码比较少？
* 有没有结合原生封装过RN组件？
* 为什么用高德地图不用百度地图？
* 怎样在Android Studio中对React Native的js代码进行断点调试？
* 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?
* 如何测试前端代码么? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?
* 前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?
* 简述一下 Handlebars 的基本用法？
* 简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？
* 用js实现千位分隔符?(来源：[前端农民工](http://div.io/topic/744)，提示：正则+replace)

function commafy(num) {  
 num = num + '';  
 var reg = /(-?d+)(d{3})/;  
  
 if(reg.test(num)){  
 num = num.replace(reg, '$1,$2');  
 }  
 return num;  
}

* 浏览器数据库 IndexedDB 入门教程

http://www.ruanyifeng.com/blog/2018/07/indexeddb.html

* 如何解决异步回调地狱

promise、generator、async/await

let hello = function(){  
 return new Promise(function(resolve,reject){  
 resolve('123');  
 });  
};  
async function say(){  
 return await hello();  
}  
say().then(function(result){  
 console.log(result);  
});

* JavaScript：彻底理解同步、异步和事件循环(Event Loop)

https://segmentfault.com/a/1190000004322358

* JavaScript 运行机制详解：再谈Event Loop

http://www.ruanyifeng.com/blog/2014/10/event-loop.html

* 网页性能管理详解

http://www.ruanyifeng.com/blog/2015/09/web-page-performance-in-depth.html

* 检测浏览器版本版本有哪些方式？

功能检测、userAgent特征检测  
  
比如：navigator.userAgent  
//"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

* 你感觉下面的代码会输出什么？

//1.考察函数提升  
var a=0;  
function b(){  
 alert(a);  
 var a= 1;  
}  
b();//undefined  
  
//2.原型的构造器  
function a(){}  
console.log(a.prototype.constructor);//self  
  
//3.this  
var a={  
 fn1:function(){  
 console.log(this);  
 }  
}  
var b=a.fn1;  
b();//window  
  
//4.typeof  
console.log(typeof "1");//string  
  
//5.闭包  
function a(){}  
function b(p){  
 var c = 0;  
 p.prototype.fn1= function(){  
 console.log(++c);  
 }  
}  
b(a);  
var d = new a();  
var e = new a();  
d.fn1();//1  
e.fn1();//2

* 为了区分对象的类型，我们用typeof操作符获取对象的类型，它总是返回一个字符串

typeof 123; // 'number'  
typeof NaN; // 'number'  
typeof 'str'; // 'string'  
typeof true; // 'boolean'  
typeof undefined; // 'undefined'  
typeof Math.abs; // 'function'  
typeof null; // 'object'  
typeof []; // 'object'  
typeof {}; // 'object'  
  
//可见，number、string、boolean、function和undefined有别于其他类型。特别注意null的类型是object，Array的类型也是object，如果我们用typeof将无法区分出null、Array和通常意义上的object——{}。  
  
//总结一下，有这么几条规则需要遵守：  
  
//不要使用new Number()、new Boolean()、new String()创建包装对象；  
  
//用parseInt()或parseFloat()来转换任意类型到number；  
  
//用String()来转换任意类型到string，或者直接调用某个对象的toString()方法；  
  
//通常不必把任意类型转换为boolean再判断，因为可以直接写if (myVar) {...}；  
  
//typeof操作符可以判断出number、boolean、string、function和undefined；  
  
//判断Array要使用Array.isArray(arr)；  
  
//判断null请使用myVar === null；  
  
//判断某个全局变量是否存在用typeof window.myVar === 'undefined'；  
  
//函数内部判断某个变量是否存在用typeof myVar === 'undefined'。  
  
//最后有细心的同学指出，任何对象都有toString()方法吗？null和undefined就没有！确实如此，这两个特殊值要除外，虽然null还伪装成了object类型。  
  
123.toString(); // SyntaxError  
  
//遇到这种情况，要特殊处理一下：  
  
123..toString(); // '123', 注意是两个点！  
(123).toString(); // '123'  
123+''; //'123'

* 个人经历面试题

1、基于flex弹性布局如何布下面的布局[A、BC内外间隔10px][B、C间隔10px]  
-----------------------  
 |--------| |--------|  
 | | | B |  
 | | |--------|  
 | A |  
 | | |--------|  
 | | | C |  
 |--------| |--------|  
-----------------------  
  
//html  
<div class="flex-container">  
 <div class="flex-item a">  
 a  
 </div>  
 <div class="flex-item b">  
 <div class="flexb-item c">  
 c  
 </div>  
 <div class="flexb-item d">  
 d  
 </div>  
 </div>  
</div>  
  
 //css  
 .flex-container{  
 display: flex;  
 height: 100vh;  
 overflow: hidden;  
 }  
 .flex-item{  
 flex: 1;  
 }  
 .flex-item.a{  
 background: #f00;  
 margin-right:5px;  
 }  
 .flex-item.b{  
 margin-left: 5px;  
 }  
  
 .flexb-item{  
 flex: 1;  
 flex-direction: column;  
 min-height: 50vh;  
 }  
 .flexb-item.c{  
 background: #c00;  
 margin-bottom: 10px;  
 }  
 .flexb-item.d{  
 background: #000;  
 }  
  
  
2、如何检测当前变量为Array？  
  
Array.isArray();  
  
3、如下两种情况分别输出什么？  
  
setTimeout(function(){  
 for(var i = 0; i<10; i++){  
 console.log(i);  
 }  
},1000);  
//0 1 2 3 4 5 6 7 8 9  
  
for(var i=0; i<10;i++){  
 setTimeout(function(){  
 console.log(i);  
 },1000);  
}  
//10 10 10 10 10 10 10 10 10 10  
  
4、如何将当前ajax查询的列表 以href的方式发给别人（拥有相同权限）,其电脑打开链接直接可以看到相同列表数据？  
  
5、是否了解Promise？  
  
[Promise](http://www.liaoxuefeng.com/wiki/001434446689867b27157e896e74d51a89c25cc8b43bdb3000/0014345008539155e93fc16046d4bb7854943814c4f9dc2000)  
  
6、前后端分离，你们团队是如何协作开发的；  
  
7、写一个todolist(Jquery、Angular、React)  
  
8、你们团队现在使用的前端框架（PC、H5）、前端自动化工具、常用库、插件，是否沉淀组件库等

## 其他问题

* 原来公司工作流程是怎么样的，如何与其他人协作的？如何跨部门合作的？
* 你遇到过比较难的技术问题是？你是如何解决的？
* 设计模式 知道什么是singleton, factory, strategy, decrator么?
* 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？
* 页面重构怎么操作？

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

* 列举IE 与其他浏览器不一样的特性？

事件不同之处：  
  
触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；  
  
获取字符代码、如果按键代表一个字符（shift、ctrl、alt除外），IE 的 keyCode 会返回字符代码（Unicode），DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；  
  
阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，需要调用 preventDefault() 方法；  
  
停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，需要调用 stopPropagation()；

* 99%的网站都需要被重构是那本书上写的？

网站重构：应用web标准进行设计（第2版)

* 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，则代码会针对旧版本的IE进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。  
如：border-shadow  
  
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版本浏览器才支持的功能,向页面增加不影响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。  
如：默认使用flash上传，但如果浏览器支持 HTML5的文件上传功能，则使用HTML5实现更好的体验；

* 是否了解公钥加密和私钥加密。

一般情况下是指私钥用于对数据进行签名，公钥用于对签名进行验证;  
HTTP网站在浏览器端用公钥加密敏感数据，然后在服务器端再用私钥解密。

* WEB应用从服务器主动推送Data到客户端有那些方式？

html5提供的Websocket  
不可见的iframe  
WebSocket通过Flash  
XHR长时间连接  
XHR Multipart Streaming  
script 标签的长时间连接(可跨域)

* 对Node的优点和缺点提出了自己的看法？

\*（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，  
因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。  
此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，  
因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。  
  
\*（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，  
而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

* 你有哪些性能优化的方法？

（看雅虎14条性能优化原则）。  
  
 （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。  
  
 （8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。  
  
 对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。

* http和https

https的SSL加密是在传输层实现的。

(1)http和https的基本概念  
http: 超文本传输协议，是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从WWW服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。  
https: 是以安全为目标的HTTP通道，简单讲是HTTP的安全版，即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。  
https协议的主要作用是：建立一个信息安全通道，来确保数组的传输，确保网站的真实性。  
(2)http和https的区别？  
http传输的数据都是未加密的，也就是明文的，网景公司设置了SSL协议来对http协议传输的数据进行加密处理，简单来说https协议是由http和ssl协议构建的可进行加密传输和身份认证的网络协议，比http协议的安全性更高。  
主要的区别如下：  
  
Https协议需要ca证书，费用较高。  
http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。  
使用不同的链接方式，端口也不同，一般而言，http协议的端口为80，https的端口为443  
http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。  
  
(3)https协议的工作原理  
客户端在使用HTTPS方式与Web服务器通信时有以下几个步骤，如图所示。  
  
客户使用https url访问服务器，则要求web 服务器建立ssl链接。  
web服务器接收到客户端的请求之后，会将网站的证书（证书中包含了公钥），返回或者说传输给客户端。  
客户端和web服务器端开始协商SSL链接的安全等级，也就是加密等级。  
客户端浏览器通过双方协商一致的安全等级，建立会话密钥，然后通过网站的公钥来加密会话密钥，并传送给网站。  
web服务器通过自己的私钥解密出会话密钥。  
web服务器通过会话密钥加密与客户端之间的通信。  
  
(4)https协议的优点  
  
使用HTTPS协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；  
HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全，可防止数据在传输过程中不被窃取、改变，确保数据的完整性。  
HTTPS是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。  
谷歌曾在2014年8月份调整搜索引擎算法，并称“比起同等HTTP网站，采用HTTPS加密的网站在搜索结果中的排名将会更高”。  
  
(5)https协议的缺点  
  
https握手阶段比较费时，会使页面加载时间延长50%，增加10%~20%的耗电。  
https缓存不如http高效，会增加数据开销。  
SSL证书也需要钱，功能越强大的证书费用越高。  
SSL证书需要绑定IP，不能再同一个ip上绑定多个域名，ipv4资源支持不了这种消耗。

* http状态码有那些？分别代表是什么意思？

简单版  
[  
 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）。  
]  
  
 完整版  
 1\*\*(信息类)：表示接收到请求并且继续处理  
 100——客户必须继续发出请求  
 101——客户要求服务器根据请求转换HTTP协议版本  
  
 2\*\*(响应成功)：表示动作被成功接收、理解和接受  
 200——表明该请求被成功地完成，所请求的资源发送回客户端  
 201——提示知道新文件的URL  
 202——接受和处理、但处理未完成  
 203——返回信息不确定或不完整  
 204——请求收到，但返回信息为空  
 205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件  
 206——服务器已经完成了部分用户的GET请求  
  
 3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理  
 300——请求的资源可在多处得到  
 301——本网页被永久性转移到另一个URL  
 302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。  
 303——建议客户访问其他URL或访问方式  
 304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用  
 305——请求的资源必须从服务器指定的地址得到  
 306——前一版本HTTP中使用的代码，现行版本中不再使用  
 307——申明请求的资源临时性删除  
  
 4\*\*(客户端错误类)：请求包含错误语法或不能正确执行  
 400——客户端请求有语法错误，不能被服务器所理解  
 401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用  
 HTTP 401.1 - 未授权：登录失败  
　　HTTP 401.2 - 未授权：服务器配置问题导致登录失败  
　　HTTP 401.3 - ACL 禁止访问资源  
　　HTTP 401.4 - 未授权：授权被筛选器拒绝  
 HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败  
 402——保留有效ChargeTo头响应  
 403——禁止访问，服务器收到请求，但是拒绝提供服务  
 HTTP 403.1 禁止访问：禁止可执行访问  
　　HTTP 403.2 - 禁止访问：禁止读访问  
　　HTTP 403.3 - 禁止访问：禁止写访问  
　　HTTP 403.4 - 禁止访问：要求 SSL  
　　HTTP 403.5 - 禁止访问：要求 SSL 128  
　　HTTP 403.6 - 禁止访问：IP 地址被拒绝  
　　HTTP 403.7 - 禁止访问：要求客户证书  
　　HTTP 403.8 - 禁止访问：禁止站点访问  
　　HTTP 403.9 - 禁止访问：连接的用户过多  
　　HTTP 403.10 - 禁止访问：配置无效  
　　HTTP 403.11 - 禁止访问：密码更改  
　　HTTP 403.12 - 禁止访问：映射器拒绝访问  
　　HTTP 403.13 - 禁止访问：客户证书已被吊销  
　　HTTP 403.15 - 禁止访问：客户访问许可过多  
　　HTTP 403.16 - 禁止访问：客户证书不可信或者无效  
 HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效  
 404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL  
 405——用户在Request-Line字段定义的方法不允许  
 406——根据用户发送的Accept拖，请求资源不可访问  
 407——类似401，用户必须首先在代理服务器上得到授权  
 408——客户端没有在用户指定的饿时间内完成请求  
 409——对当前资源状态，请求不能完成  
 410——服务器上不再有此资源且无进一步的参考地址  
 411——服务器拒绝用户定义的Content-Length属性请求  
 412——一个或多个请求头字段在当前请求中错误  
 413——请求的资源大于服务器允许的大小  
 414——请求的资源URL长于服务器允许的长度  
 415——请求资源不支持请求项目格式  
 416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段  
 417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。  
  
 5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求  
 HTTP 500 - 服务器遇到错误，无法完成请求  
 　　HTTP 500.100 - 内部服务器错误 - ASP 错误  
 　　HTTP 500-11 服务器关闭  
 　　HTTP 500-12 应用程序重新启动  
 　　HTTP 500-13 - 服务器太忙  
 　　HTTP 500-14 - 应用程序无效  
 　　HTTP 500-15 - 不允许请求 global.asa  
 　　Error 501 - 未实现  
 HTTP 502 - 网关错误  
 HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常  
  
  
HTTP协议的状态消息都有哪些?  
“200〃 : OK（成功） 一切正常  
“302〃 : Found（临时移动）类似于301，但新的URL应该被视为临时性的替代，而不是永久性的。  
“400〃 : Bad Request（错误请求）请求出现语法错误。  
“404〃 : Not Found（未找到）无法找到指定位置的资源。  
“500〃 : Internal Server Error（服务器内部错误）服务器遇到错误，无法完成请求。  
“502〃 : Bad Gateway（错误网关）服务器作为网关或者代理时，  
为了完成请求访问下一个服务器，但该服务器返回了非法的应答。

* 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？（流程说的越详细越好）

注：这题胜在区分度高，知识点覆盖广，再不懂的人，也能答出几句，  
 而高手可以根据自己擅长的领域自由发挥，从URL规范、HTTP协议、DNS、CDN、数据库查询、  
 到浏览器流式解析、CSS规则构建、layout、paint、onload/domready、JS执行、JS API绑定等等；  
  
 详细版：  
1、浏览器会开启一个线程来处理这个请求，对 URL 分析判断如果是 http 协议就按照 Web 方式来处理;  
2、调用浏览器内核中的对应方法，比如 WebView 中的 loadUrl 方法;  
3、通过DNS解析获取网址的IP地址，设置 UA 等信息发出第二个GET请求;  
4、进行HTTP协议会话，客户端发送报头(请求报头);  
5、进入到web服务器上的 Web Server，如 Apache、Tomcat、Node.JS 等服务器;  
6、进入部署好的后端应用，如 PHP、Java、JavaScript、Python 等，找到对应的请求处理;  
7、处理结束回馈报头，此处如果浏览器访问过，缓存上有对应资源，会与服务器最后修改时间对比，一致则返回304;  
8、浏览器开始下载html文档(响应报头，状态码200)，同时使用缓存;  
9、文档树建立，根据标记请求所需指定MIME类型的文件（比如css、js）,同时设置了cookie;  
10、页面开始渲染DOM，JS根据DOM API操作DOM,执行事件绑定等，页面显示完成。  
  
 简洁版：  
浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；  
服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；  
浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）；  
载入解析到的资源文件，渲染页面，完成。

* 部分地区用户反应网站很卡，请问有哪些可能性的原因，以及解决方法？
* 从打开app到刷新出内容，整个过程中都发生了什么，如果感觉慢，怎么定位问题，怎么解决?
* 除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？
* 你常用的开发工具是什么，为什么？

Sublime Text 3 + 相关插件编写前端代码  
Google chrome 、Mozilla Firefox浏览器 +firebug 兼容测试和预览页面UI、动画效果和交互功能  
Node.js+Gulp  
git 用于版本控制和Code Review

* 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。

* 你怎么看待Web App 、hybrid App、Native App？
* 你移动端前端开发的理解？（和 Web 前端开发的主要区别是什么？）
* 加班的看法？

加班就像借钱，原则应当是------救急不救穷

* 如何开发和部署前端代码？
* web前后端如何分离？
* 前端如何与后端协作？
* 如何理解前端模块化、组件化？
* 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 命民以该JS 功能为准英文翻译；  
  
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

* 如何设计突发大规模并发架构？
* 说说最近最流行的一些东西吧？常去哪些网站？

ES6\WebAssembly\Node\MVVM\Web Components\React\React Native\Webpack 组件化

* 当团队人手不足，把功能代码写完已经需要加班的情况下，你会做前端代码的测试吗？
* 知道什么是SEO并且怎么优化么? 知道各种meta data的含义么?
* 移动端（Android IOS）怎么做好用户体验?

清晰的视觉纵线、信息的分组、极致的减法、  
利用选择代替输入、标签及文字的排布方式、  
依靠明文确认密码、合理的键盘利用、

* 简单描述一下你做过的移动APP项目研发流程？
* 你在现在的团队处于什么样的角色，起到了什么明显的作用？
* 你认为怎样才是全端工程师（Full Stack developer）？
* 介绍一个你最得意的作品吧？
* 你有自己的技术博客吗，用了哪些技术？
* 对前端安全有什么看法？
* 是否了解Web注入攻击，说下原理，最常见的两种攻击（XSS 和 CSRF）了解到什么程度？
* 项目中遇到国哪些印象深刻的技术难题，具体是什么问题，怎么解决？。
* 最近在学什么东西？
* 你的优点是什么？缺点是什么？
* 如何管理前端团队?
* 最近在学什么？能谈谈你未来3，5年给自己的规划吗？
* 想问公司的问题？

问公司问题：  
目前关注哪些最新的Web前端技术（未来的发展方向）？  
前端团队如何工作的（实现一个产品的流程）？  
公司的薪资结构是什么样子的？

## 优质网站推荐

1. 极客标签： http://www.gbtags.com/
2. 码农周刊： http://weekly.manong.io/issues/
3. 前端周刊： http://www.feweekly.com/issues
4. 极客头条： http://geek.csdn.net/
5. Startup News：http://news.dbanotes.net/
6. Hacker News： https://news.ycombinator.com/news
7. InfoQ： http://www.infoq.com/
8. w3cplus： http://www.w3cplus.com/
9. Stack Overflow： http://stackoverflow.com/
10. Atp： http://atp-posts.b0.upaiyun.com/posts/

了解更多：http://note.youdao.com/share/?id=f179a18d162ce08afa020a8ba8fc0cdf&type=note

## 须掌握的基础知识点

作为一名前端工程师，**无论工作年头长短都应该必须掌握的知识点**：

此条由 王子墨 发表在 [前端随笔](http://julying.com/blog/front-end-engineers-must-master-knowledge/)

1、DOM结构——两个节点之间可能存在哪些关系以及如何在节点之间任意移动。

document.documentElement 返回文档的根节点<html>  
document.body <body>  
document.activeElement 返回当前文档中被击活的标签节点(ie)  
event.fromElement 返回鼠标移出的源节点(ie)  
event.toElement 返回鼠标移入的源节点(ie)  
event.srcElement 返回激活事件的源节点(ie)  
event.target 返回激活事件的源节点(firefox)  
当前对象为node  
返回父节点：node.parentNode, node.parendElement,  
返回所有子节点：node.childNodes（包含文本节点及标签节点）,node.children  
返回第一个子节点：node.firstChild  
返回最后一个子节点： node.lastChild  
返回同属上一个子节点：node.nextSibling  
返回同属下一个子节点：node.previousSibling  
parentNode和parentElement功能一样，childNodes和children功能一样。但是parentNode和  
childNodes是符合W3C标准的，可以说比较通用。而另外两个只是IE支持，不是标准，Firefox就不支持  
,所以大家只要记得有parentElement和children就行了

2、DOM操作——怎样添加、移除、移动、复制、创建和查找节点。

（1）创建新节点  
 createDocumentFragment() //创建一个DOM片段  
 createElement() //创建一个具体的元素  
 createTextNode() //创建一个文本节点  
（2）添加、移除、替换、插入  
 appendChild()  
 removeChild()  
 replaceChild()  
 insertBefore()  
（3）查找  
 getElementsByTagName() //通过标签名称  
 getElementsByName() //通过元素的Name属性的值  
 getElementById() //通过元素Id，唯一性

3、事件——怎样使用事件以及IE和DOM事件模型之间存在哪些主要差别。

（1）冒泡型事件：事件按照从最特定的事件目标到最不特定的事件目标(document对象)的顺序触发。  
IE 5.5: div -> body -> document  
IE 6.0: div -> body -> html -> document  
Mozilla 1.0: div -> body -> html -> document -> window  
（2）捕获型事件(event capturing)：事件从最不精确的对象(document 对象)开始触发，  
然后到最精确(也可以在窗口级别捕获事件，不过必须由开发人员特别指定)。  
（3）DOM事件流：同时支持两种事件模型：捕获型事件和冒泡型事件，  
但是，捕获型事件先发生。两种事件流会触及DOM中的所有对象，从document对象开始，也在document对象结束。  
DOM事件模型最独特的性质是，文本节点也触发事件(在IE中不会)。

4、XMLHttpRequest——这是什么、怎样完整地执行一次GET请求、怎样检测错误。

//MLHttpRequest 对象提供了在网页加载后与服务器进行通信的方法。  
  
<script type="text/javascript">  
var xmlhttp;  
function loadXMLDoc(url){  
 xmlhttp=null;  
 if(window.XMLHttpRequest){//code for all new browsers  
 xmlhttp=newXMLHttpRequest();  
 }elseif(window.ActiveXObject){//code for IE5 and IE6  
 xmlhttp=newActiveXObject("Microsoft.XMLHTTP");  
 }  
 if(xmlhttp!=null){  
 xmlhttp.onreadystatechange=state\_Change;  
 xmlhttp.open("GET",url,true);  
 xmlhttp.send(null);  
 }else{  
 alert("Your browser does not support XMLHTTP.");  
 }  
}  
  
function state\_Change(){  
 if(xmlhttp.readyState==4){ //4 = "loaded"  
 if(xmlhttp.status==200){ //200 = OK  
 //...our code here...  
 }else{  
 alert("Problem retrieving XML data");  
 }  
 }  
}  
</script>

5、严格模式与混杂模式——如何触发这两种模式，区分它们有何意义。

在标准模式中，浏览器根据规范呈现页面；  
在混杂模式中，页面以一种比较宽松的向后兼容的方式显示。  
浏览器根据DOCTYPE是否存在以及使用的哪种DTD来选择要使用的呈现方法。  
如果XHTML文档包含形式完整的DOCTYPE，那么它一般以标准模式呈现。  
对于HTML 4.01文档，包含严格DTD的DOCTYPE常常导致页面以标准模式呈现。  
包含过渡DTD和URI的DOCTYPE也导致页面以标准模式呈现，但是有过  
渡DTD而没有URI会导致页面以混杂模式呈现。DOCTYPE不存在或形式不正确会导致HTML和XHTML文档以混杂模式呈现。

6、盒模型——外边距、内边距和边框之间的关系，IE 8以下版本的浏览器中的盒模型有什么不同。

一个元素盒模型的层次从内到外分别为：内边距、边框和外边距  
IE8以下浏览器的盒模型中定义的元素的宽高不包括内边距和边框。（这句回答有争议，期待能有人解答原因QQ1662484899）  
  
延伸：  
盒子模型分为两类：W3C标准盒子模型和IE盒子模型 （微软确实不喜欢服从他家的标准）  
这两者的关键差别就在于：  
W3C盒子模型——属性高（height）和属性宽（width）这两个值不包含 填充（padding）和边框（border）  
IE盒子模型——属性高（height）和属性宽（width）这两个值包含 填充（padding）和边框（border）  
我们在编写页面代码的时候应该尽量使用标准的W3C盒子模型（需要在页面中声明DOCTYPE类型），  
这样可以避免多个浏览器对同一页面的不兼容。  
因为如果不声明DOCTYPE类型，IE会将盒子模型解释为IE盒子模型，FireFox等会将其解释为W3C盒子模型；  
而如果在页面中声明了DOCTYPE模型，所有的浏览器都会把盒子模型解释为W3C盒子模型。

7、块级元素与行内元素——怎么用CSS控制它们、它们怎样影响周围的元素以及你觉得应该如何定义它们的样式。

块级元素，用CSS中的display:inline;属性则变为行内元素  
行内元素，用CSS中的display:block;属性则变为块级元素  
影响：周围元素显示在同一行或换行显示，根据具体情况调整样式

8、浮动元素——怎么使用它们、它们有什么问题以及怎么解决这些问题。

需要浮动的元素可使用CSS中float属性来定义元素的浮动位置，left：往左浮动，right：往右浮动  
浮动元素引起的问题：  
（1）父元素的高度无法被撑开，影响与父元素同级的元素  
（2）与浮动元素同级的非浮动元素会跟随其后  
（3）若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构  
解决方法：  
使用CSS中的clear:both;属性来清除元素的浮动可解决2、3问题，  
对于问题1，添加如下样式，给父元素添加clearfix样式：  
.clearfix:after{content: ".";display: block;height: 0;clear: both;visibility: hidden;}  
.clearfix{display: inline-block;} /\* for IE/Mac \*/

9、HTML与XHTML——二者有什么区别，你觉得应该使用哪一个并说出理由。

主要区别：  
XHTML 元素必须被正确地嵌套  
XHTML 元素必须被关闭，空标签也必须被关闭，如 <br> 必须写成 <br />  
XHTML 标签名必须用小写字母  
XHTML 文档必须拥有根元素  
XHTML 文档要求给所有属性赋一个值  
XHTML 要求所有的属性必须用引号""括起来  
XHTML 文档需要把所有 < 、>、& 等特殊符号用编码表示  
XHTML 文档不要在注释内容中使“--”  
XHTML 图片必须有说明文字  
XHTML 文档中用id属性代替name属性

10、JSON——它是什么、为什么应该使用它、到底该怎么使用它，说出实现细节来。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。  
易于人阅读和编写。同时也易于机器解析和生成。  
JSON建构于两种结构：  
“名称/值”对的集合（A collection of name/value pairs）。  
不同的语言中，它被理解为对象（object），纪录（record），结构（struct），  
字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。  
值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

## 文档推荐

1. [jQuery 基本原理](http://docs.huihoo.com/jquery/jquery-fundamentals/zh-cn/index.html)
2. [JavaScript 秘密花园](http://bonsaiden.github.io/JavaScript-Garden/zh/)
3. [CSS参考手册](http://css.doyoe.com/)
4. [JavaScript 标准参考教程](http://javascript.ruanyifeng.com/)
5. [ECMAScript 6入门](http://es6.ruanyifeng.com/)

## 额外推荐：

* 事先声明： ◆ 这些题目的来源：面试曾经被问过;工作被别人问过或者遇见过;网上看见过... ◆ 答案真心不给提供，真的是许多问题都需要个人的理解和沉淀， 所以还请各位自己动手...前端可以试试自己差不多能回答多少题，哈哈。 ◆ 其实很多题我也没有好答案，面试的时候如果遇到牛人我也可以顺便交流交流， 反正我也只是一面，不丢人也不怕丢人。 ◆ 如果朋友们有好的面试题欢迎提建议，我会其实补充更新的，先谢谢各位了。
* HTML相关
* 1. <!DOCTYPE>标签的定义与用法。  
  2. 块级元素和行内元素都有哪些?  
  3. 你真的了解HTML吗? 雅虎面试题　　把前面黄底那段拿去搜索下就知道了(曾在某浪公司面试的时候被问到过，确实是很好的问题)。  
  CSS相关  
  1. 介绍所知道的CSS hack技巧(如：\_， \*， +， \9， !important 之类)。  
  2. 介绍CSS盒模型。  
  3. CSS层叠是什么?介绍一下。  
  4. 都知道哪些CSS浏览器兼容性问题。  
  5. 有时会被问到些刁钻点的题，比如position值都有哪些，CSS3都有哪些新内容...  
  JavaScript基础相关  
  1. HTTP协议的状态消息都有哪些?(如200、302对应的描述)  
  2. AJAX是什么? AJAX的交互模型(流程)? AJAX跨域的解决办法?  
  3. 同步和异步的区别?  
  4. 简述JavaScript封装。  
  5. JavaScript继承有哪两种形式形式，进行描述。  
  6. 什么是闭包?以下代码点击<p> 会输出什么?为什么?能大概说明白的话继续问能想出几种解决办法。  
  <!DOCTYPE HTML>  
  <html>  
  <head>  
  <meta charset="utf-8" />  
  <title>闭包演示</title>  
  <style type="text/css">  
   p {background:gold;}  
  </style>  
  <script type="text/javascript">  
  function init() {  
   var pAry = document.getElementsByTagName("p");  
   for( var i=0; i<pAry.length; i++ ) {  
   pAry[i].onclick = function() {  
   alert(i);  
   }  
   }  
  }  
  </script>  
  </head>  
  <body onload="init();">  
  <p>产品 0</p>  
  <p>产品 1</p>  
  <p>产品 2</p>  
  <p>产品 3</p>  
  <p>产品 4</p>  
  </body>  
  </html>  
  7. 在JS中this关键字的使用场合和用法(如在构造函数中、setTimeout中等)。  
  8. 简述下cookie的操作，还有cookie的属性都知道哪些。  
  9. IE与FF的JS兼容性都知道哪些。  
  10. DOM操作 - 怎样添加、移除、移动、复制、创建和查找节点(这个问题真心是基础题，一般不会问)。  
  jQuery相关  
  1. jQuery源码是否尝试去读过?说说基本的架构或者 jQuery.fn.init 中都做了哪些判断。  
  2. 都知道哪些不好的jQuery书写方式。  
  3. Sizzle是否有读过?  
  其它相关的加分项：  
  1. 都使用和了解过哪些编辑器?都使用和了解过哪些日常工具?  
  2. 都知道有哪些浏览器内核?开发过的项目都兼容哪些浏览器?  
  3. 国内外的JS牛人都知道哪些?  
  4. 瀑布流布局或者流式布局是否有了解  
  4. 正则表达式有系统学习过吗(看书或网上教程)?有的话就问问简单点的邮箱验证、URL验证， 或者问问 贪婪匹配与懒惰匹配 的理论知识。  
  5. Node.js是否有过尝试?到什么程度?说说个人理解的看法?  
  6. HTML5都有哪些新的JS API?  
  7. 前端优化知识都知道哪些?  
  8. 基础算法题(如快速排序，能否一两句说说重要的核心原理或者数组消重等)。
* http://www.ruanyifeng.com/blog/2011/04/quicksort\_in\_javascript.html "快速排序"的思想很简单，整个排序过程只需要三步： （1）在数据集之中，选择一个元素作为"基准"（pivot）。 　　（2）所有小于"基准"的元素，都移到"基准"的左边；所有大于"基准"的元素，都移到"基准"的右边。 　　（3）对"基准"左边和右边的两个子集，不断重复第一步和第二步，直到所有子集只剩下一个元素为止。
* var quickSort = function(arr) {  
   //小于等于1直接返回  
   　　if (arr.length <= 1) { return arr; }  
   //获取最中间的序号  
   　　var pivotIndex = Math.floor(arr.length / 2);  
   //获取最中间的那一项  
   　　var pivot = arr.splice(pivotIndex, 1)[0];  
   　　var left = [];  
   　　var right = [];  
   //循环匹配  
   　　for (var i = 0; i < arr.length; i++){  
   　　　　if (arr[i] < pivot) {  
   　　　　　　left.push(arr[i]);  
   　　　　} else {  
   　　　　　　right.push(arr[i]);  
   　　　　}  
   　　}  
   　　return quickSort(left).concat([pivot], quickSort(right));  
  };
* 9. 是否有接触过或者了解过重构。  
  原文：http://www.cnblogs.com/Darren\_code/archive/2012/01/31/questions.html
* 10.tcp三次握手，一句话概括
* 客户端和服务端都需要直到各自可收发，因此需要三次握手。
  1. TCP和UDP的区别
  + （1）TCP是面向连接的，udp是无连接的即发送数据前不需要先建立链接。  
    （2）TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付。 并且因为tcp可靠，面向连接，不会丢失数据因此适合大数据量的交换。  
    （3）TCP是面向字节流，UDP面向报文，并且网络出现拥塞不会使得发送速率降低（因此会出现丢包，对实时的应用比如IP电话和视频会议等）。  
    （4）TCP只能是1对1的，UDP支持1对1,1对多。  
    （5）TCP的首部较大为20字节，而UDP只有8字节。  
    （6）TCP是面向连接的可靠性传输，而UDP是不可靠的。
* 【编辑推荐】

[什么是Node.js？](http://developer.51cto.com/art/201109/288849.htm)

[前端必读：浏览器内部工作原理](http://developer.51cto.com/art/201202/314376.htm)

[HTML 5 APIs程序员指南](http://developer.51cto.com/art/201202/314376.htm)

[10个在线正则表达式测试网站推荐](http://developer.51cto.com/art/201103/251633.htm)

[从新浪微博的改版谈网页重构](http://developer.51cto.com/art/201109/290414.htm)