

目录

1.HTML.....	10
1. HTML5 特性有哪些？语义化的做法有哪些，具体指的什么？	10
2. 你了解浏览器标准模式和怪异模式吗？	11
3. XHTML 与 HTML 有什么区别？	11
4. 为什么要在标签上使用 data-，它有什么好处了？	12
5. Meta 标签，都有一些什么特性，有什么作用？	12
6. 什么是 Canvas，你使用它做个什么需求？	12
7. HTML 废弃的标签有哪些？举几个例子（拓展型题）！	13
8. 工作中你都兼容哪些浏览器？实际开发中你都遇到过哪些兼容问题？	13
9. CSS / JS 引入的位置一般在哪里？为什么	14
10. 什么是渐进式渲染？	14
11. HTML 模板语言有哪些？	14
12. Meta viewport 的原理是什么？	15
2.CSS.....	15
1. 盒模型组成是什么？，box-sizing 有什么作用？	15
2. CSS3 新特性有哪些，伪类，伪元素，锚伪类分别有哪些？	15
3. 使用 CSS 实现隐藏元素的方式有几种？	17
4. 如何实现盒子在页面水平和垂直居中？	17
5. 详细的描述一下 position，display？	17
6. 如何实现盒子在页面水平和垂直居中？	18
7. 详细的描述一下 position，display？	18
8. 请解释 *{box-sizing:border-box;} 的作用，并说明使用它的好处？	19
9. 浮动元素引起的问题和解决办法？	19

10.	Link 和 @import 引入 CSS 的区别 ?	20
11.	解释一下 css3 的 flexbox, 以及适用场景 ?	20
12.	inline 和 inline-block 的区别 ?	20
13.	哪些是块级元素那些是行级元素, 各有什么特点 ?	21
14.	网格布局了解吗 (grid) ?	22
15.	传统表格布局了解吗? (table) ?	22
16.	实现两栏布局你知道有哪些方法?	22
17.	DPI 是什么?	22
18.	Attribute / property 有什么区别了?	23
19.	css 布局问题? css 实现三列布局怎么做? 如果中间是自适应又怎么做?	23
20.	流式布局如何实现, 响应式布局如何实现?	23
21.	三栏布局 (圣杯布局 / 双飞翼 / Flex 布局) ?	23
22.	transition 和 animation 的区别, animation 的属性有什么, 如何实现加速度重力模拟?	23
23.	请写出使用 CSS3 旋转图片的代码片段?	错误! 未定义书签。
24.	Less / sass 使用过么?	25
25.	对移动端开发了解多少? (响应式设计、Zepto; @media、viewport、JavaScript 正则表达式判断平台。)	26
26.	什么是 BFC,如何去创建 BFC ? 它能解决什么问题 ?	26
27.	CSS 中的长度单位有哪些 (px ,pt,rem,em,ex,vw,vh,vmin,vmax) ?	26
28.	CSS 中选择器的优先级关系?	27
29.	雪碧图 (精灵图) ?	27
30.	什么是 SVG,它能做什么?	27
31.	媒体查询的原理是什么?	27
32.	CSS 的加载时异步的吗? 它表现在什么地方?	28

33.	工作中遇到的浏览器兼容问题有哪些？你都知道哪些常用的 HACK 技巧？	28
34.	请解释一下 “::before” 和 “:after” 中的双冒号和单冒号的区别？	29
3.JS.....		29
1.	js 的基本类型有哪些？引用类型有哪些？ null 和 undefined 的区别。	29
2.	如何判断一个变量是 Array 类型？如何判断一个变量是 Number 类型？（都不止一种）。	30
3.	Object 是引用类型嘛？引用类型和基本数据类型有什么区别？堆栈关系了解吗？	30
4.	JS 常见的 DOM 操作 API？	31
5.	解释一下事件冒泡和事件捕获？	32
6.	事件委托（手写例子），事件冒泡和捕获，如何阻止冒泡？如何阻止默认事件？	32
7.	对闭包的理解？什么时候构成闭包？闭包的实现方法？闭包的优缺点？	33
8.	this 有哪些使用场景？跟 C, JAVA 中的 this 有什么区别？如何改变 this 的指向？	34
9.	call, apply, bind 有什么区别？	34
10.	显示原型和隐式原型，手绘原型链，原型链是什么？为什么要有原型链？	35
11.	你知道哪几种创建对象的方式？	35
12.	实现继承有哪些方式，他们的优缺点是什么？	35
13.	New 一个对象具体做了什么？	36
14.	请写出 AJAX，或者说出流程？（答案有问题）	36
15.	变量提升如何理解？	37
16.	举例说明一个匿名函数的典型用例？	37
17.	指出 JS 的宿主对象和原生对象的区别？为什么扩张 JS 内置对象是不提倡的做法？有哪些内置对象和函数举几个例子吧。	37
18.	Attribute 和 property 之间的区别是什么？	38
19.	Document load 和 document DOMContentLoaded 两个事件之前的区别？	38
20.	=== 和 == , [] === [], undefined === undefined, [] == [], undefined == undefined.....	38

21.	Typeof 能够得到哪些值？	38
22.	什么是 “use strict” ,好处和坏处是什么？	39
23.	函数的作用域是什么？ JS 的作用域你了解吗？	39
24.	Js 如何实现重载和多态？	40
25.	常用的数组 API，字符串 API 有哪些？	40
26.	原生事件绑定（跨浏览器），下面案例中 dom0 与 dom1 的区别是？	41
27.	给定一个元素获取它相对于视图窗口的坐标？	41
28.	Js 的字符串类型有哪些方法？正则表达式的函数怎么使用？	42
29.	深拷贝原理是什么，请简要描述过程？	43
30.	你能编写一个可复用的事件监听方法吗？	44
31.	浏览器（web 端）cookie 如何设置或获取？	45
32.	setTimeout 和 promise 的执行顺序是什么？	45
33.	javaScript 的事件流模型都有什么？	45
34.	navigator 对象，location 和 history 你了解多少？	46
35.	js 的垃圾回收机制（了解）？	47
36.	说一说内存泄漏的原因和场景？	47
37.	Dom 事件中的 target 和 currentTarget 有什么区别？	47
38.	Typeof 和 instanceof 区别，instanceof 的原理是什么？	47
39.	JS 动画和 CSS3 动画有什么区别？	48
40.	JS 实现一个简单的倒计时程序？	48
41.	JS 处理异常的语句有哪些举例说明？	48
42.	JS 的设计模式你知道哪些？	49
43.	轮播图组件开发，我有 10000 万张图片要轮播？	50
44.	说一说 websocket 的工作原理和机制？	50

45.	手指点击触控屏幕是什么事件，函数柯里化是什么？	51
46.	JS 代码调试？	51
4.	ES6.....	52
1.	聊聊 promise？	52
2.	ES6 特性你了解多少？如果遇到一个东西不知道是 ES6 还是 ES5，你该如何区分？	52
3.	ES6 的继承和 ES5 的继承有什么区别？	53
4.	Promise 如何封装一个 AJAX？	53
5.	let 和 const 的优点？	54
6.	ES6 generator 是什么，async / await 实现原理？	54
7.	ES6 和 node 的 commonjs 模块化规范的区别。	54
8.	箭头函数，以及他们的 this。	54
5.	计算机网络相关.....	54
1.	HTTP 协议头包含哪些重要部分，HTTP 状态码。	54
2.	网络 url 输入到输出都做了什么？。	55
3.	为什么说性能优化就要减少 HTTP 的访问次数？。	56
4.	描述一下 HTTP 的请求过程与原理？。	56
5.	https 有几次握手和挥手？https 的原理什么是？。	56
6.	http 有几次握手和挥手？TLS 的中文名？TLS 在那一网络层？。	57
7.	TCP 连接的特点，TCP 连接如何保证安全可靠？	57
8.	为什么 TCP 连接需要三次握手？。	57
9.	为什么 TCP 连接需要三次握手四次挥手？。	57
10.	TCP 的三次握手和四次挥手绘图（当场画写 ACK 和 SEQ 的值）？。	58
11.	TCP 与 UDP 的区别有哪些？	59
12.	Get 和 Post 的区别？什么情况下用到？	59

13.	HTTP2 / HTTP1 之间的区别是什么? 。	60
14.	介绍一下 websocket? 。	60
15.	HTTP Response 的 Header 里面都有什么? 。	61
6.	浏览器相关	61
1.	为什么会有跨域的问题出现? 。	61
2.	前端安全 XSS,CSRF 这些是什么? 。	62
3.	浏览器如何加载页面的, script 脚本阻塞有什么解决办法, defer 和 async 的区别是什么? 。	62
4.	浏览器强制缓存和协商缓存是什么? 。	63
5.	浏览器的全局变量有哪些? 。	63
6.	浏览器同一时间能够从一个域名下载多少个资源? 。	63
7.	按需加载, 不同页面的元素判断标准是怎么样的? 。	63
8.	WEB 存储, COOKIES,LOCALSTOGE 等的使用规则和区别? 。	64
9.	浏览器内核你都知道哪些? 。	64
10.	什么是预加载, 懒加载? 。	64
11.	一个 XMLHttpRequest 实例它有几种状态分别代表什么? 。	65
12.	DNS 解析原理, 输入网址之后如何查找服务器? 。	65
13.	服务器如何识别是你在操作, 说说思路? 。	65
14.	浏览器的渲染流程你了解吗? 。	66
15.	介绍几个 IE 浏览器的兼容问题? 。	66
16.	对于 Session 你知道哪些内容? 。	67
17.	如何实现一个拖拽, 说一下思路? (表述方式需要修改) 。	67
18.	拆解一下 URL 的各个部分, 分别是什么意思? 。	68
7.	前端工程化常用手段	68
1.	Webpack,gulp,grunt 等构建工具了解多少, 它们有什么区别? 。	68

2. Webpack,的入口文件如何配置？	69
3. Webpack 的 loader 和 plugins 的区别？	70
8 模块化相关.....	70
1. 对 AMD,CMD,CommonJs 有什么理解？	70
2. 为什么要模块化？ 不用的时候和用 RequireJS 的时候代码该如何书写？	73
3. 分别说说同步和异步模块化的应用场景，说一下 AMD 异步模块化实现的原理？	73
9. 框架 / 类库.....	73
1. 说一说你都使用了什么框架？	73
2. zepto 和 jquery 是什么关系，他们之间有什么联系？	73
3. jquery 源码如何实现选择器，为什么\$取得对象要设计成数组的形式，这样设计有什么目的了？	73
4. jquery 如何绑定事件，有几种类型和区别？	74
5. 什么是 MVVM,MVC,MVP？	74
6. Vue 和 Angular 的双向数据绑定原理？	75
7. Vue 和 Angular 的组件通信以及路由原理？	76
10. NodeJs.....	79
1. 对 node.js 有没有了解.....	79
2. Express 和 koa 有什么关系，有什么区别.....	79
3. node.js 适合做什么业务.....	79
4. node.js 与 php 和 Java 的区别.....	80
5. Nodejs 中的 Stream 和 Buffer 有什么区别.....	80
6. node 的异步问题是如何解决的.....	80
11. 数据结构相关.....	81
1. 基本数据构：（数组、 对列、 链表、堆、二叉树、嘻哈表等等）	81
2. 8 种排序算法，原理，以及适用的场景和复杂度。	83

3. 说出越多越好的费波拉切数列的实现方法？	85
12. 性能优化相关	86
1. CDN 的用法是什么？ 什么时候用到	86
2. CDN 的用法是什么？ 什么时候用到	86
3.如何优化 dom 操作的性能。	88
4.单页面应用有什么 SEO 方案？ 。	89
5. 单页面应用有什么 SEO 方案？ 。	89
13. 其他相关（ 拔高 ）	90
1.正则表达式	90
2.前端渲染和后端渲染的区别	91
3.数据库的四大特性， 什么是原子性， 表的关系	92
4.你觉得前端体系应该是怎样的	94
6. SEO	94
7. mysql 和 mongoDB 有什么区别	95
8. click 在 ios 上有 300ms 延迟， 原因如何及如何解决。	97
9. 移动端的适配， rem+媒体查询/meta 头设置。	98
10. 移动端的手势和事件。	99
11. unicode, utf8, gbk 编码的了解， 乱码的解决。	101
14. 技术拓展加分（ 拔高 ）	103
1. 你都看过什么书？ 最近在看什么书。	103
2. 用过什么框架？ 有没有看过什么框架的代码。	103
3. 有没有学过设计模式。	103
4. 说一说观察者模式。	104
4. 你最大的优点是什么？ 那你最大的缺点呢？	105

5.平时看哪些博客.....	106
6. 现在你的领导给你了一份工作，要求你一个星期完成，但你看了需求以后估计需要 3 周才能完成，你该怎么办？	106
7. 平时关注的前端技术.....	106
8. 如何规划自己的职业生涯.....	107
9. 项目过程中，有遇到什么问题吗？怎么解决的.....	107
10. 最近在研究哪方面的东西.....	107
11. 请介绍一项你最热爱、最擅长的专业领域，并且介绍的学习规划.....	108
12. 请介绍你参与的印象最深刻的一个项目，为什么？并且介绍你在项目中的角色和发挥的作用。	109
15. HR 面试的开放性问题（体现积极向上的人生观）	109
1.你为什么要学习前端。	109
2. 你平时的是怎么学习前端的？有什么输出。	110
3.你觉得自己最好的项目是什么。	110
4. 身边比较佩服的人有什么值得你学习的？你为什么没有跟他们一样。	110
5. 同事的什么问题会让你接受不了.....	110
6. 压力最大的事情是什么.....	111
7. 身边的朋友通常对你的评价是什么.....	111
8. 喜欢什么样的工作氛围.....	111
8. 如何看待加班.....	112
9. 如何看待加班.....	112
10. 未来职业规划.....	112
11. 未来职业规划.....	113
12. 你现在手里有没有 offer。	113

1.HTML

1. HTML5 特性有哪些？语义化的做法有哪些，具体指的什么？

新特性常用解释如下：


1	多媒体，用于媒介回放的 video 和 audio 元素。
2	图像效果，用于绘画的 canvas 元素，svg 元素等。
3	离线&存储，对本地离线存储能够更好地支持，比如 localStorage,Cookies 等。
4	设备兼容特性，HTML5 提供了前所未有的数据与应用接入开放接口，使外部应用可以直接与浏览器内部的数据直接相连。
5	连接特性，更有效的连接工作效率，使得基于页面的实时聊天，更快速的网页游戏体验，更优化的在线交流得到了实现，同时拥有更有效的服务器推送技术，Server-Sent Event 和 WebSockets 就是其中的两个特性，这两个特性能够帮助我们实现服务器将数据“推送”到客户端的功能。
6	性能与集成特性，HTML5 会通过 XML HttpRequest2 等技术，帮助您的 Web 应用和网站在多样化的环境中更快速地工作。

新增标签（ 常用 ）：

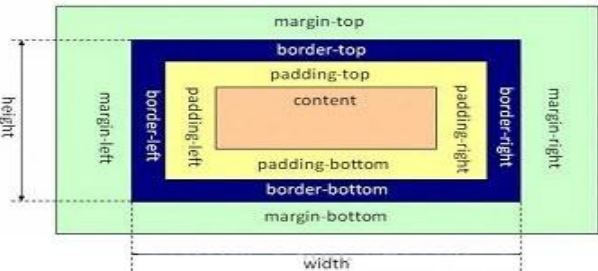
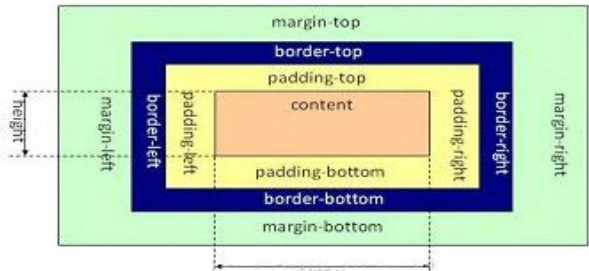
1	多媒体：<audio> <video> <source><embed><track>
2	新表单元素：<datalist> <output> <keygen>
3	新文档节段和纲要：<header>页面头部 <section>章节 <aside>边栏<article>文档内容 <footer>页面底部

语义化具体指以下内容：

1	语义特性，添加<header></header><nav></nav>等标签
2	为了在没有 css 代码时，也能呈现很好的内容结构，代码结构，以至于达到没有编程基础的非技术人员，也能看懂其代码
3	为了提高用户体验，比如：title,alt 用于解释名词和图片信息
4	利于 SEO，语义化能和搜索引擎建立良好的关系，有利于爬虫抓取更多的有效信息，爬虫依赖于标签来确定上下文和各个关键字的权重

5	便于团队开发和维护，语义化更具可读性，如果遵循 W3C 标准的团队都遵循这个标准，可以减少差异化，有利于规范化
6	方便其他设备解析（如屏幕阅读器，盲人阅读器，移动设备）以语义的方式来渲染网页
 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！	

2. 你了解浏览器标准模式和怪异模式吗？

<p>产生原因：现代的浏览器一般都有两种渲染模式：标准模式和怪异模式，在标准模式下，浏览器按照 HTML 和 CSS 标准对文档进行解析和渲染；而在怪异模式下，浏览器则按照旧有的非标准的实现方式对文档进行解析和渲染，这样的话，对于旧有的网页，浏览器就会启动怪异模式，就能够使得旧网页正常显示；对于新的网页，则可以启动标准模式，使得新网页能够使用 HTML 和 CSS 的标准特性。</p>	
<p>标准模式</p> 	<p>怪异模式</p> 
给 span 等行内元素设置 width 和 height 无效。	然而我，可以生效！
margin:0 auto 水平居中，在标准模式下没有问题。	用 margin:0 auto 设置水平居中在怪异模式下会失效，解决办法，用 text-align 属性来解决
右边的这些问题，在标准模式下均不会出现问题！	怪异模式下设置图片的 padding 会失效，怪异模式下 Table 中的字体属性不能继承上层的设置怪异模式下 white-space:pre 会失效

 提示：回答该问题，主要理解出现这种模式的原因，然后记忆 3-5 条区别即可。

3. XHTML 与 HTML 有什么区别 ？

1：XHTML 元素必须被正确的嵌套。
2：XHTML 元素必须被关闭。

3: 标签名必须用小写字母。

4: XHTML 文档必须拥有根元素。

! 提示: 所以记住几条吧老铁, 这个问题基础哟。

4. 为什么要在标签上使用 data-, 它有什么好处了 ?

1: html5 规范中规定自定义属性需要添加前缀 data-, 目的是提供与渲染无关的信息

2: 使用 `getAttribute()` 方法以及 `setAttribute()` 方法操作自定义属性

3: 使用 `dataset` 操作自定义属性。

4: `dataset` 属性的兼容性问题, Chrome 8+ Firefox(Gecko) 6.0+ Internet Explorer 11+ Opera11.10+ Safari 6+ 以上浏览器均能很好支持, 想要知道更多的兼容信息, 可以访问右边的二维码。

5: 使用 `data-` 属性选择器好处: 绑定 DOM 强相关数据, js 传递数据



! 提示: 右边二维码可以查看许多关于兼容问题的数据!

5. Meta 标签, 都有一些什么特性, 有什么作用 ?

1: 什么是 meta 标签: `<meta>` 标签提供关于 HTML 文档的元数据, 它不会显示在页面上, 但是对于机器是可读的, 可用于浏览器 (如何显示内容或重新加载页面), 搜索引擎 (关键词), 或其他 web 服务。

2: meta 的作用: meta 里的数据是供机器解读的, 告诉机器该如何解析这个页面, 还有一个用途是可以添加服务器发送到浏览器的 HTTP 头部内容。

3: 常用 meta 标签总结: Charset: 它是声明文档使用的字符编码, 以防乱码, 而且一定要写在第一行 Viewport: 主要是影响移动端页面布局的。

! 提示: 所以阅读 3 遍心里有个印象, 能说出 2 点即可!

6. 什么是 Canvas, 你使用它做个什么需求?


什么是 Canvas: Canvas 元素是 HTML5 的一部分, 允许脚本语言动态渲染位图像。Canvas 由一个可绘制地区 HTML 代码中的属性定义决定高度和宽度。JavaScript 代码可以访问该地区, 通过一套完整的绘图功能类似于其他通用二维的 API, 从而生成动态的图形。

Canvas 能应对以下需求：

1. **游戏**：毫无疑问，游戏在 HTML5 领域具有举足轻重的地位。HTML5 在基于 Web 的图像显示方面比 Flash 更加立体、更加精巧，Ohad 认为运用 Canvas 制作的图像能够令 HTML5 游戏在流畅度和跨平台方面发挥更大的潜力。

2. **图表制作**：图表制作时常被人们忽略，但无论企业内部还是企业间交流合作都离不开图表。现在一些开发者使用 HTML/CSS 完成图标制作，但完全可以用 Canvas 来实现。当然，使用 SVG（可缩放矢量图形）来完成图表制作也是非常好的方法。（例如：echarts.js heightchart.js 都是基于 canvas 来绘图!）

3. **banner 广告**：Flash 曾经辉煌的时代，智能手机还未曾出现。现在以及未来的智能机时代，HTML5 技术能够在 banner 广告上发挥巨大作用，用 Canvas 实现动态的广告效果再合适不过。

 **提示**：其实还有很多领域（例如：模拟器 / 远程计算机控制 / 字体设计 / 图像编辑器 / 其它），但是就当前来说，基于 canvas 图表制作是我们必须要掌握的能力。

7. HTML 废弃的标签有哪些？举几个例子（拓展型题）！

因为,,<ins>,这些标签有语意，所以是,<i>,<u>,<s>的替代品。

=<i>	定义强调的文字。
<ins>=<u>	定义插入的文字。
=<s>	定义删除的文字。
=	定义重要性强调的文字。

 **提示**：总之丢掉的东西必然是因为不适合了，记住几个即可！

8. 工作中你都兼容哪些浏览器？实际开发中你都遇到过哪些兼容问题？

关于浏览器兼容的这个问题太笼统了 所以当面试官问这个问题时你可以问一下面试官你指的是哪方面。

现在就就一些常见的 css 样式

Input 的按钮在 ios 和 Android 中的样式兼容问题

ios 有默认的渲染 方式 渐变和圆角而且他的行高并不会随着内容的增加。

解决方法：

在标签上增加

-webkit-appearance: none;

Height: 4rem;

滚动穿透的问题:

滚动穿透是指在移动端当有 fixed 遮罩和弹出层的时候，屏幕上的滑动能够滑动背景下的内容。

最后的解决方法

```
body.modal-open{  
    position:fixed;  
    width:100%;  
}
```


 提示：理解内容即可！

9. CSS / JS 引入的位置一般在哪里？为什么

Script 标签最好放在</body>标签的前面，因为放在所有 body 中的标签后面就不会出现网页加载时出现空白的情况，可以持续的给用户提供视觉反馈，同时在有些情况下，会降低错误的发生；而 css 标签应该放在<head></head>标签之间，因为如果放在</body>标签的前面，那么当 DOM 树构建完成了，渲染树才构建，那么当渲染树构建完成，浏览器不得不再重新渲染整个页面，这样造成了资源的浪费，效率也不高，如果放在<head></head>之间，浏览器边构建边渲染，效率要高得多 ！

 提示：理解内容即可！

10. 什么是渐进式渲染 ？

 指打开页面先加载首屏显示的内容，之后再随着时间或者滚动页面才进行后面的加载 。

11. HTML 模板语言有哪些 ？

Django 模板基于 Python 的一个模板。

Python 下有许多款不同的 Web 框架。Django 是重量级选手中最有代表性的一位。许多成功的网站和 APP 都基于 Django。

Django 是一个开放源代码的 Web 应用框架，由 Python 写成。

Django 遵守 BSD 版权，初次发布于 2005 年 7 月，并于 2008 年 9 月发布了第一个正式版本 1.0 。

Django 采用了 MVC 的软件设计模式，即模型 M，视图 V 和控制器 C。

12. Meta viewport 的原理是什么？

手机浏览器是把页面放在一个虚拟的“窗口”（viewport）中，通常这个虚拟的“窗口”（viewport）比屏幕宽，这样就不用把每个网页挤到很小的窗口中（这样会破坏没有针对手机浏览器优化的网页的布局），用户可以通过平移和缩放来看网页的不同部分。移动设备默认的 viewport 是 layout viewport，也就是那个比屏幕要宽的 viewport，但在进行移动设备网站的开发时，我们需要的是 ideal viewport。

2.CSS

1. 盒模型组成是什么？， box-sizing 有什么作用？


盒模型：margin(外边距) padding(内边距) border(边框) content(内容)

box-sizing:content-box / border-box / inherit 有以下三个属性

1：content-box:指定元素的宽度和高度（最小/最大属性）适用于 box 的宽度和高度，元素的填充和边框布局和绘制指定宽度和高度除外。

2：border-box:指定宽度和高度（最小/最大属性）确定元素边框 box, 对元素指定宽度和高度包括 padding 和 border 的指定，内容的宽度和高度减去各自双方该边框和填充的宽度从指定的“宽度”和“高度”属性计算

3：inherit:指定 box-sizing 属性的值，应该从父元素继承

 **提示：该内容理解为主，盒模型以及属性需要记忆！**

2. CSS3 新特性有哪些，伪类，伪元素，锚伪类分别有哪些？

C3 新特性：

- | | |
|---|--|
| 1 | CSS 实现圆角（border-radius）,阴影（box-shadow）,边框图片 border-image |
| 2 | 对文字加特效（text-shadow）,强制文本换行（word-wrap）,线性渐变（linear-gradient） |
| 3 | 旋转，缩放，定位，倾斜：transform:rotate(90deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg) |
| 4 | 增加了更多的 CSS 选择器，多背景，rgba(); |

5	在 CSS3 中唯一引入的伪元素是::selection;
6	媒体查询 (@media) ,多栏布局 (flex)
伪类：用于向某些选择器添加特殊的效果	
1	: hover 将样式添加到鼠标悬浮的元素
2	: active 将样式添加到被激活的元素
3	: focus 将样式添加到获得焦点的元素
4	: link 将样式添加到未被访问过的链接
5	: visited 将样式添加到被访问过的链接
6	: first-child 将样式添加到元素的第一个子元素
7	: lang 定义指定的元素中使用的语言
伪元素：用于将特殊的效果添加到某些选择器，伪元素代表了某个元素的子元素，这个子元素虽然在逻辑上存在，但却并不实际存在于文档树中	
1	::first-letter 将样式添加到文本的首字母
2	::first-line 将样式添加到文本的首行
3	::before 在某元素之前插入某些内容
4	::after 在某元素之后插入某些内容
新增伪类：	
1	P:first-of-type 选择属于其父元素的首个<p>元素的每个<p>元素
2	P:last-of-type 选择属于其父元素的最后<p>元素的每个<p>元素
3	P:only-of-type 选择属于其父元素唯一的<p>元素的每个<p>元素
4	P:only-child 选择属于其父元素唯一的子元素的每个<p>元素
5	P:nth-child(n) 选择属于其父元素的第 n 个子元素的每个<p>元素
6	P:nth-last-child(n) 选择属于其父元素的倒数第 n 个子元素的每个<p>元素
7	P:nth-of-type(n) 选择属于其父元素第 n 个<p>元素的每个<p>元素
8	P:nth-last-of-type(n) 选择属于其父元素倒数第 n 个<p>元素的每个<p>元素
9	P:last-child 选择属于其父元素最后一个子元素的每个<p>元素
10	P:empty 选择没有子元素的每个<p>元素(包括文本节点)
11	P:target 选择当前活动的<p>元素
12	:not(p) 选择非<p>元素的每个元素
13	:enabled 控制表单控件的可用状态

3. 使用 CSS 实现隐藏元素的方式有几种 ？

1	Opacity:设置一个元素的透明度 .hide{opacity:0;}
2	Visibility .hide{visibility:hidden}
3	Display:确保元素不可见并且连盒模型也不生成 .hide{display:none}
4	Position .hide{position:absolute; top:-9999px; left:-9999px;}
5	Clip-path .hide{clip-path:polygon(0px 0px, 0px 0px, 0px 0px, 0px 0px);}



提示：该内容需要记忆！

4. 如何实现盒子在页面水平和垂直居中 ？

水平居中设置：

1	行内元素：设置 text-align:center;
2	定宽块状元素：设置 左右 margin 值为 auto;
3	不定宽块状元素
4	在元素外加入 table 标签(完整的，包括 table,tbody,tr,td),该元素写在 td 内，然后设置 margin 的值为 auto;
5	给该元素设置 display:inline 方法；
6	父元素设置 position:relative 和 left:50%,子元素设置 position:relative 和 left:50%

垂直居中设置：

1	父元素高度确定的单行文本，设置 height=line-height;
	父元素高度确定的多行文本
2	A:插入 table（插入方法和水平居中一样）,然后设置 vertical-align:middle; B:先设置 display:table-cell 再设置 vertical-align:middle;

5. 详细的描述一下 position , display ？

1	Opacity:设置一个元素的透明度 .hide{opacity:0;}
2	Visibility .hide{visibility:hidden}


3	Display:确保元素不可见并且连盒模型也不生成 .hide{display:none}
4	Position .hide{position:absolute; top:-9999px; left:-9999px;}
5	Clip-path .hide{clip-path:polygon(0px 0px, 0px 0px, 0px 0px, 0px 0px);}
 提示：该内容需要记忆！	

6. 如何实现盒子在页面水平和垂直居中 ？

水平居中设置：	
1	行内元素：设置 text-align:center;
2	定宽块状元素：设置 左右 margin 值为 auto;
3	不定宽块状元素
4	在元素外加入 table 标签(完整的，包括 table,tbody,tr,td),该元素写在 td 内，然后设置 margin 的值为 auto;
5	给该元素设置 display:inline 方法;
6	父元素设置 position:relative 和 left:50%,子元素设置 position:relative 和 left:50%
垂直居中设置:	
1	父元素高度确定的单行文本，设置 height=line-height;
2	父元素高度确定的多行文本 A:插入 table（插入方法和水平居中一样）,然后设置 vertical-align:middle; B:先设置 display:table-cell 再设置 vertical-align:middle;

7. 详细的描述一下 position , display ？

position	
position: absolute;	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
position:fixed;	生成绝对定位的元素，相对于浏览器窗口进行定位.
position:relative;	生成相对定位的元素，相对于其正常位置进行定位
position:static;	默认值，没有定位，元素在正常的文档流(忽略 top,bottom,left,right 或者 z-index 声明)
position:inherit;	规定应该从父元素继承 position 属性的值

display: (经常使用的)	
display:none	此元素不会被显示
display:block;	此元素将显示为块级元素，此元素前后会带有换行符
display:inline;	行内块元素
display:inline-block;	此元素会作为列表显示
 提示： 基础知识，必须要了解！	

8. 请解释 `*{box-sizing:border-box;}` 的作用，并说明使用它的好处 ？

它是 CSS3 新增的属性：

通常一个块级元素实际所占的宽高度=margin+(border-width)+padding+height/width;如果设置了 border-box，实际所占的宽高度=设置的高度/设置的宽度+margin 无论如何改动 border 和 padding 的值，都不会导致 box 总尺寸发生变化，也就不会打乱页面整体布局


提示： 该内容需要理解记忆！

9. 浮动元素引起的问题和解决办法？

浮动会引起以下问题：	
1	由于浮动元素已脱离文档流，所以父元素无法被撑开，影响与父级元素同级的元素
2	与浮动元素同级的非浮动元素（内联元素）会跟随其后，也是由于浮动元素脱离文档流，不占据文档流中的位置
3	如果该浮动元素不是同级第一个浮动的元素，则他之前的浮动也应该浮动，否则容易影响页面的结构显示
清除浮动的常用方法：（ 其实百度一大堆 ， 关键需要记忆和实际操作 ）	
1	额外标签法，江湖人称 “隔墙法” 。
2	使用伪类 <code>after</code> 等等，其实是隔墙法的变异操作。
3	浮动外部元素。

4 设置高度或者使用 overflow:hidden 或者 auto.



提示：如果有人说你连清除浮动都不会，这将是来自宇宙级别的讽刺。想更深入了解？BFC了解一下。

10. Link 和 @import 引入 CSS 的区别？

适用范围不同 @import 可以在网页页面中使用，也可以在 css 文件中使用，用来将多个 CSS 文件引入到一个 CSS 文件中；而 link 只能将 CSS 文件引入到网页页面中。

功能范围不同 link 属于 XHTML 标签，而@import 是 CSS 提供的一种方式，link 标签除了可以加载 CSS 外，还可以定义 RSS,定义 rel 连接属性等，@import 就只能加载 CSS

加载顺序不同 当一个页面被加载的时候，Link 引用的 CSS 会同时被加载，而@import 引用的 CSS 会等到页面全部被下载完再被加载，所以有时候浏览@import 加载 CSS 的页面时开始会没有样式（就是闪烁），这种是在网速慢的时候才会看得出来

兼容性差别 由于@import 是 CSS2.1 提出的，所以老的浏览器不支持，而@import 只有在 IE5 以上的才能识别，而 link 标签无此问题

控制样式时差别 使用 Link 方式可以让用户切换 CSS 样式



提示：理解记忆，

11. 解释一下 css3 的 flexbox，以及适用场景？

弹性布局的主要思想是让容器有能力来改变项目的宽度和高度，以填满可用空间（主要是为了容纳所有类型的显示设备和屏幕尺寸）的能力，flexbox 布局是最合适的一个应用程序的组件，以及小规模布局，而网络布局是用于较大规模的布局




提示：用自己话去描述最佳！

12. inline 和 inline-block 的区别？

inline


1	inline 元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化
2	Inline 元素设置 width,height 属性无效
3	inline 元素的 margin 和 padding 属性，水平方向的 padding-left,padding-right,margin-left,margin-right 都产生边距效果；但垂直方向的 padding-top,padding-bottom,magin-top,margin-bottom 不会产生边距效果。
Inline-block	
简单来说就是将对象改变成 Inline 对象，但是对象的内容作为 block 对象呈现，之后的内联对象会被排列在同一行内	
 基础知识啦，一定要了解！	

13. 哪些是块级元素那些是行级元素，各有什么特点？

行内元素	块级元素
a b em font i img input br label span small select textarea	address dl div form h1-h6 hr menu ol p
行内元素不会独占一行，相邻的行内元素会排列在同一行里，一行排不下会自动换行，其宽度随元素的内容而变化。	块级元素会独占一行，其宽度自动填满其父元素宽度
块级元素可以设置 margin,padding,行内元素的水平方向的 padding-left，padding-right，margin-left，margin-right 都产生边距效果，但是垂直方向 padding-top,padding-bottom,margin-top,margin-bottom 都不会产生边距效果	块级元素可以设置 width,height 属性，行内元素设置宽高无效
 非常基础的知识。强化记忆哟！	


14. 网格布局了解吗 （ grid ）？

Grid 布局又称“网格”，是一个基于二维网格布局的系统，主要目的是改变我们基于网格设计的用户接口方式，Grid 布局是第一个专门为解决布局问题而创建的 CSS 模块

 提示：可以结合官方文档了解一下，可以了解一下 bootstrap 里的常用组件


15. 传统表格布局了解吗？（ table ）？

Table 用于布局，主要用来显示批量的数据；但是 table 有一个缺点，就是加载页面的时候，需要将全部的数据都请求到，才显示页面，否则就是一片空白，所以现在一般都不用 table 进行布局.但因其稳定性非常优秀，一般在管理系统类项目中应用较多。

 提示：分析几种常见的布局方式，看看他们的区别，需要了解

16. 实现两栏布局你知道有哪些方法？

1	margin-bottom 和 padding-bottom
2	table-row 和 table-cell
3	flex 布局

 提示：方法有很多，基本会带出以上三种即可！

17. DPI 是什么？

Dpi:每英寸包含点的数量(dots per inch)

普通屏幕通常包含 96dpi,一般将 2 倍于此的屏幕称之为高分屏,即大于等于 192dpi 的屏幕,比如 Mac 视网膜屏就达到了 192dpi(即 2dppx),打印时一般会需要更大的 dpi;

1dppx=96dpi	1dpi 约等于 0.39dpcm	1dpcm 约等于 2.54dpi
1in =2.54cm=25.4mm=101.6q=72pt=6pc=96px;		
支持 IE9+ firefox3.5+ chrome29.0+		



提示：移动端开发如果不清楚 DPI，将是非常尴尬的事！

18. Attribute / property 有什么区别了？

Property	它是 DOM 中的属性，是 javascript 里边的对象
Attribute	它是 HTML 标签上的特性，它的值只能够是字符串

19. css 布局问题？css 实现三列布局怎么做？如果中间是自适应又怎么做？

20. 流式布局如何实现，响应式布局如何实现？

流式布局	响应式布局
也叫 fluid,当上面一行的空间不够容纳新的 TextView 时候才开辟下一行的空间，场景：主要用于关键词搜索或者热门标签等场景；他主要是按照页面元素的宽度按照屏幕分辨率进行适配调整，但整体布局不变,使用百分比定义宽度，高度大都是用 px 来固定，可以根据可视区域和父元素的实时尺寸来调整，尽可能适应各种分辨率	主要是实现不同屏幕分辨率的终端上浏览网页的不同展示方式，通过响应式设计能使网站在手机和平板电脑上有更好的浏览阅读体验；首先设置 meta 标签，通过媒体查询来设置样式 Media Queries,然后再设置多种试图宽度



提示：这两个问题是前端面试布局方便的万金油，一定要了解哟！

21. 三栏布局（圣杯布局 / 双飞翼 / Flex 布局）？

22. transition 和 animation 的区别，animation 的属性有什么，如何实现加速度重力模拟？

transiton: 过渡属性 过渡所需要时间 过渡动画函数 过渡延迟时间；

transition 的优点在于简单易用，但是它有几个很大的局限。

1	transition 需要事件触发，所以没法在网页加载时自动发生
---	----------------------------------

2	transition 是一次性的，不能重复发生，除非一再触发
3	transition 只能定义开始状态和结束状态，不能定义中间状态，也就是说只有两个状态
4	一条 transition 规则，只能定义一个属性的变化，不能涉及多个属性
5	animation-name: none 为默认值，将没有任何动画效果，其可以用来覆盖任何动画
6	animation-duration: 默认值为 0，意味着动画周期为 0，也就是没有任何动画效果
7	animation-timing-function: 与 transition-timing-function 一样
8	animation-delay: 在开始执行动画时需要等待的时间
9	animation-iteration-count: 定义动画的播放次数，默认为 1，如果为 infinite，则无限次循环播放
10	animation-direction: 默认为 normal，每次循环都是向前播放，（0-100），另一个值为 alternate，动画播放为偶数次则向前播放，如果为基数词就反方向播放
11	animation-state: 默认为 running，播放，paused，暂停
12	animation-fill-mode: 定义动画开始之前和结束之后发生的操作，默认值为 none，动画结束时回到动画没开始时的状态；forwards，动画结束后继续应用最后关键帧的位置，即保存在结束状态；backwards，让动画回到第一帧的状态；both: 轮流应用 forwards 和 backwards 规则

CSS3 的 animation 制作动画效果主要包括两部分：

1. 用关键帧声明一个动画


2. 在 animation 调用关键帧声明的动画。@keyframes 就是关键帧。这个帧与 Flash 里的帧类似，一个动画中可以有很多个帧。一个@keyframes 中的样式规则是由多个百分比构成的，可以在这个规则上创建多个百分比，从而达到一种不断变化的效果。另外，@keyframes 必须要加 webkit 前缀

区别:animation 属性类似于 transition，他们都是随着时间改变元素的属性值，其主要区别在于：transition 需要触发一个事件才会随着时间改变其 CSS 属性；animation 在不需要触发任何事件的情况下，也可以显式的随时间变化来改变元素 CSS 属性，达到一种动画的效果

重力模拟实现

从物理课本中我们知道，物体掉落时受重力加速度 g 的影响而加速掉落，当接触地面时到达最大速度 v_{max} 然后回弹，

回弹到达最大高度时速度为 0。由于动能损失，每次回弹的高度会越来越小，直到最后静止。已知值确定：由于是动画过程，所以总时间 t 我们也应该知道；起始下落高度 $_h1$ ，动能损失过程复杂，为了简单实现效果，所以我们做如下规定：弹跳次数为 3 次，回弹高度我们确定为 $_h2=_h1/7$ ， $_h3=_h1/35$ ， $_h4=_h1/105$

 提示：了解为主，内容太多。一定要看看！

23. 请写出使用 CSS3 旋转图片的代码片段？



(扫码查看详情)

24. Less / sass 使用过么？

LESS	SASS
是一门 css 预处理语言，他扩充了 css 语言，增加了诸如变量，混合(mixin),函数等功能，让 css 更易维护，方便制作主题，扩充，less 可以运行在 Node 或浏览器端	是一种 css 的开发工具，提供了许多便利的写法，大大节省了设计者的时间，使得 css 的开发，变得简单和可维护

 提示：工具类型，理解即可！

25. 对移动端开发了解多少？（响应式设计、Zepto；@media、viewport、JavaScript 正则表达式判断平台。）

26. 什么是 BFC,如何去创建 BFC？它能解决什么问题？

BFC: 简单说，它是一种属性，这种属性会影响着元素的定位以及与其兄弟元素之间的相互作用

如何创建 BFC：当一个 HTML 元素满足下面条件的任何一点，都可以产生 BFC,常见的，float 的值不为 'none'，overflow 的值不为 'visible'，display 的值为 'table-cell'，'table-caption'，or 'inline-block' 中的任何一个，position 的值不为 'static' 或 'relative' 中的任何一个创建一个 BFC 的元素就是一个独立的盒子，里面的子元素不会在布局上影响外面的元素，反之亦然，同时 BFC 仍然属于文档中的普通流

 **BFC 是一个非常重要的概念。一定要了解哟！**

27. CSS 中的长度单位有哪些（px,pt,rem,em,ex,vw,vh,vmin,vmax）？

绝对长度单位：

PX	实际上是一个按角度度量的单位，是一个像素单位
IN	英寸是一个物理度量单位 1in==96px
Cm	厘米是比较熟悉有用的物理度量单位，他也映射成像素 1cm==37.8px
Mm	毫米是个小数量级的物理度量单位 1mm==0.1cm==3.78px

相对字体的长度：

Em	是一个相对单位 1em==16px==0.17in==12pt==1pc==4.2mm==0.42cm
Rem	rem 和 em 一样也是一个相对单位，但是和 em 不同的是 rem 总是相对于根元素，而不像 em 一样使用级联的方式来计算尺寸
Point	它是一个物理度量单位 1pt=1/72in
Pica	pica 和 points 一样，只不过 1pc=12pt

Ex	ex 是一个基于当前字体的 x 字母高度度量的
Ch	ch 的内涵和 x 高度相似，只是 ch 是基于字符 0 的宽度的而不是基于字符 x 高度的
可视区百分比长度单位：	
Vw	是可视区宽度单位，1vw 等于可视区宽度的百分之一，vw 单位跟百分比很相似
 提示：特别是 PX / REM /EM 将在项目中应用	

28. CSS 中选择器的优先级关系？

! Important>行内样式>ID 选择器>类选择器>标签>通配符>继承>浏览器默认属性

29. 雪碧图 （ 精灵图 ）？

是把网站上用到的一些图片整合到一张单独的图片中，从而减少你的网站的 HTTP 请求数量，该图片使用 css background 和 background-position 属性渲染，这也就意味着你的标签变得更复杂了，图片是在 css 中定义，并非在标签中，优点：减少加载网页图片时对服务器的请求次数，可以合并多数背景图片和小图标，方便在任何位置使用，这样不同位置的请求只需要调用一个图片，从而减少对服务器的请求次数，降低服务器压力，同时提高了页面的加载速度，节约服务器的流量。

30. 什么是 SVG,它能做什么？

SVG 它是可伸缩矢量图形，用来定义用于网络的基于矢量的图形，图形在放大或者改变尺寸的情况，其图形质量不会有所损失。

 提示：矢量图和位图的区别也就在于矢量图无限放大不影响图片质量。

31. 媒体查询的原理是什么？

H5 的新特性，为了移动端的使用而新增的特性，使用 @media 查询，你可以针对不同的媒体类型定义不同的样式

32. CSS 的加载时异步的吗？它表现在什么地方？

1	触发异步样式下载的诀窍是使用一个 <link> 元素，并 为 media 属性设置一个不可用的值（我用的是 media=" none"，不过其它的任何值也是可以的）。当一个媒体查询的结果值计算出来是 false 的时候，浏览器仍然会下载样式表，但是不会在渲染页面之前等待样式表的资源可用
2	延迟添加 link 标签到页面，这种方式比较兼容可以使用同样原理，延迟加载 JS

33. 工作中遇到的浏览器兼容问题有哪些？你都知道哪些常用的 HACK 技巧？

浏览器兼容问题		
1：不同浏览器的默认初始值不同。		
解决方法：引入全局控制样式，或者使用第三方		
2：块属性标签 float 后，又有横行的 margin 的情况下，在 IE6 显示 margin 比设置的大。		
解决方法：在 float 的标签样式控制中加入 display:inline,将其转化为行内属性。		
3：设置小于 10px 高度标签，在 IE6,7,无法正常显示设置高度。		
解决方法：给超出高度的标签设置 overflow:hidden;或者设置行高 line-height 小于你设置的高度		
4：行内属性标签，设置 display:block 后采用 float 布局，又有横行的 margin 的情况，IE6 间距 bug		
解决方法：在 display:block;后面加入 display:inline;display:table;		
5：图片默认有间距。		
解决方法：使用 float 属性为 img 布局。		
6：标签最低高度设置 min-height 不兼容。		
解决方法：如果我们要设置一个标签的最小高度 200px，需要进行的设置为：{min-height:200px; height:auto !important; height:200px; overflow:visible;}		
常用的 HACK 技巧		
le 浏览器 hack 如下		
_nowamagic:1px;-----ie6	*nowamagic:1px;-----ie7	nowamagic:1px\0;-----ie89

```
nowamagic:1px\9\0;-----ie9 :root nowamagic:1px; -----ie9
```

Firefox 与 Chrome 的 Hack

```
Firefox:@-moz-document url-prefix() /*写在选择器外层时（只可写在此处）: Firefox only*/
```

```
Chrome:@media screen and (-webkit-min-device-pixel-ratio:0) /*写在选择器外层时（只可写在此处）: Chrome only*/
```

 提示：非常重要了，所谓的工作经验就是解决问题的能力。这些方法你一定能用的到！

34. 请解释一下“::before”和“:after”中的双冒号和单冒号的区别？

如果你的网站只需要兼容 webkit、firefox、opera 等浏览器，建议对于伪元素采用双冒号的写法，如果不得不兼容 IE 浏览器，还是用 CSS2 的单冒号写法比较安全

3.JS

1. js 的基本类型有哪些？引用类型有哪些？null 和 undefined 的区别。

基本数据类型

undefined

null

boolean

number

string

引用数据类型

function

object

array

Undefined:表示变量声明但未初始化时的值

null 表示准备用来保存对象，还没有真正保存对象的值。从逻辑角度看，null 值表示一个空对象指针。

说明

JavaScript（ECMAScript 标准）里共有 5 种基本类型：Undefined, Null, Boolean, Number, String，和一种复杂类型 Object。可以看到 null 和 undefined 分属不同的类型，未初始化定义的值用 typeof 检测出来是"undefined"(字符串)，而 null 值用 typeof 检测出来是"object"（字符串）。任何时候都不建议显式的设置一个变量为 undefined，但是如果保存对象的变量还没有真正保存对象，应该设置成 null。实际上，undefined 值是派生自 null 值的，ECMAScript 标准规定对二者进行相等性测试要返回 true

2. 如何判断一个变量是 Array 类型？如何判断一个变量是 Number 类型？ （ 都不止一种 ）。

从原型入手，Array.prototype.isPrototypeOf(obj);也可以从构造函数入手，obj instanceof Array 根据对象的 class 属性(类属性)，跨原型链调用 toString()方法。Array.isArray()方法。

isNaN () 是一个函数，用 isNaN 判断一个变量，返回一个 Boolean 值。若返回的值为 false，则为可以转换成数字类型；返回的值是 true，则不能转换成数字类型

Typeof () 判断

3. Object 是引用类型嘛？引用类型和基本数据类型有什么区别？堆栈关系了解吗？ 。

Object 是引用类型。

基本类型：1.基本类型的值是不可变得，2.基本类型的比较是值的比较。3.基本类型的变量是存放在栈区的（栈区指内存里的栈内存）。

引用类型：1.引用类型的值是可变的，2.引用类型的值是同时保存在栈内存和堆内存中的对象。

引用类型与基本类型比较

基本类型：string,number,boolean,null,undefined		引用类型：Function,Array,Object	
访问方式			
操作和保存在变量的实际的值		存在内存中，js 不许直接访问内存，操作的是对象的引用	
存储的位置			
保存在栈区		引用存放在栈区，实际对象保存在堆区。	

复制前的变量对象

num1	5 (Number 类型)

复制后的变量对象

num2	5 (Number 类型)
num1	5 (Number 类型)

复制前的变量对象

obj1	(Object 类型)

复制后的变量对象

obj2	(Object 类型)
obj1	(Object 类型)

堆内存

Object

Object

Object

图 4-1

图 4-2

4. JS 常见的 DOM 操作 API?

节点查找API

`document.getElementById`：根据ID查找元素，大小写敏感，如果有多个结果，只返回第一个；
`document.getElementsByClassName`：根据类名查找元素，多个类名用空格分隔，返回一个 `HTMLCollection`。注意兼容性为IE9+（含）。另外，不仅仅是`document`，其它元素也支持 `getElementsByClassName` 方法；
`document.getElementsByTagName`：根据标签查找元素，* 表示查询所有标签，返回一个 `HTMLCollection`。
`document.getElementsByName`：根据元素的`name`属性查找，返回一个 `NodeList`。
`document.querySelector`：返回单个`Node`，IE8+(含)，如果匹配到多个结果，只返回第一个。
`document.querySelectorAll`：返回一个 `NodeList`，IE8+(含)。
`document.forms`：获取当前页面所有form，返回一个 `HTMLCollection`；

节点创建API

`createElement`创建元素：

```
var elem = document.createElement("div");
elem.id = 'haorooms';
elem.style = 'color: red';
elem.innerHTML = '我是新创建的haorooms测试节点';
document.body.appendChild(elem);
```

通过 `createElement` 创建的元素并不属于 `document` 对象，它只是创建出来，并未添加到html文档中，要调用 `appendChild` 或 `insertBefore` 等方法将其添加到HTML文档中。

`createTextNode`创建文本节点：

```
var node = document.createTextNode("我是文本节点");
document.body.appendChild(node);
```

`cloneNode` 克隆一个节点：

`node.cloneNode(true/false)`，它接收一个bool参数，用来表示是否复制子元素。

```
var from = document.getElementById("test");
var clone = from.cloneNode(true);
clone.id = "test2";
document.body.appendChild(clone);
```

克隆节点并不会克隆事件，除非事件是用

这种方式绑定的，用 `addEventListener` 和 `node.onclick=xxx` 方式绑定的都不会复制。

`createDocumentFragment`

本方法用来创建一个 `DocumentFragment`，也就是文档碎片，它表示一种轻量级的文档，主要是用来存储临时节点，大量操作DOM时用它可以大大提升性能。



提示：这类知识点网络上已经有非常成熟的总结，详情扫描此二维码：



5. 解释一下事件冒泡和事件捕获？

事件冒泡：	当你使用事件捕获时，父级元素先触发，子级元素后触发。
事件捕获：	当你使用事件冒泡时，子级元素先触发，父级元素后触发。

6. 事件委托（手写例子），事件冒泡和捕获，如何阻止冒泡？如何阻止默认事件？

实例如下：

```
1 var toolbar = document.querySelector(".toolbar");
2
3 toolbar.addEventListener("click", function (e) {
4     var button = e.target;
5     if (!button.classList.contains("active"))
6         button.classList.add("active");
7     else
8         button.classList.remove("active");
9 });
10
```

事件冒泡，就是元素自身的事件被触发后，如果父元素有相同的事件，如 onclick 事件，那么元素本身的触发状态就会传递，也就是冒到父元素，父元素的相同事件也会一级一级根据嵌套关系向外触发，直到 document/window，冒泡过程结束。

但是事件冒泡在某些应用场景产生一些问题，就是我们不需要触发的事件，由于冒泡的原因，也会运行。所以在这个时候要取消事件冒泡。**阻止事件冒泡如下：**

```
1 box.onmouseover = function (event) {
2     // 阻止冒泡
3     event = event || window.event;
4     if (event && event.stopPropagation) {
5         event.stopPropagation();
6     } else {
7         event.cancelBubble = true;
8     }
9 }
10
```

事件捕获，与事件冒泡相反，事件会从最外层开始发生，直到最具体的元素。事件捕获的概念下发生 click 事件的顺序应该是 document -> html -> body -> div -> p。**阻止事件冒泡如下：**


```
1 //阻止浏览器的默认行为
2 function stopDefault(e) {
3     //阻止默认浏览器动作(W3C)
4     if (e && e.preventDefault)
5         e.preventDefault();
6     //IE中阻止函数器默认动作的方式
7     else
8         window.event.returnValue = false;
9     return false;
10 }
11 |
```

7. 对闭包的理解？什么时候构成闭包？闭包的实现方法？闭包的优缺点？

函数内部可以直接读取全局变量，但是在函数外部无法读取函数内部的局部变量。闭包就是能够读取其他函数内部变量的函数。内部函数对外部函数的变量有了引用关系——闭包就是这时产生的。每次对外部函数的调用，都会产生一次闭包。

实现方法：

1 给函数添加一些属性

2 声明一个变量，将一个函数当做值赋给变量

3 new 一个对象，然后给对象添加属性和方法

4 var obj={}就是声明一个空的对象

用处

它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中，不会在 f1 调用后被自动清除。

(1) 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在 IE 中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

(2) 闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

8. this 有哪些使用场景？跟 C, JAVA 中的 this 有什么区别？如何改变 this 的指向？

this 的使用场景

- 1 全局&调用普通函数，在全局环境中，this 永远指向 window
- 2 构造函数，如果函数作为构造函数使用，那么其中的 this 就代表它即将 new 出来的对象
- 3 对象方法，如果函数作为对象的方法时，方法中的 this 指向该对象。注意：若是在对象方法中定义函数，那么情况就不同了。函数 a 虽然是在 b 内部定义的,但它仍然属于一个普通函数，this 仍指向 window
- 4 构造函数 prototype 属性，即便是在整个原型链中，this 代表的也是当前对象的值
- 5 函数用 call、apply 或者 bind 调用，当一个函数被 call、apply 或者 bind 调用时，this 的值就取传入的对象
- 6 DOM event this，前六种情况其实可以总结为：this 指向调用该方法的对象
- 7 箭头函数中的 this，当使用箭头函数的时候，情况就有所不同了：箭头函数内部的 this 是词法作用域，由上下文确定

java 中 this.value 可以再本类中调用全局变量,也可以在构造器中用 this()调用其他构造器,也可以用 this 表示当前对象 JavaScript 中 this 指的是这个函数所属的对象的值,当 new 一个函数时,这个 this 就会指向这个 new 出来的对象,apply()和 call()可以改变一个函数中 this 指向的对象 call 和 apply 都可以改变 this 指向，不过 call 的第二个参数是散列分布，apply 则可以是一个数组

9. call,apply,bind 有什么区别？

call() 和 apply()的第一个参数相同，就是指定的对象。这个对象就是该函数的执行上下文。

call()在第一个参数之后的 后续所有参数就是传入该函数的值。

apply() 只有两个参数，第一个是对象，第二个是数组，这个数组就是该函数的参数。

bind() 方法和前两者不同在于：bind() 方法会返回执行上下文被改变的函数而不会立即执行，而前两者是直接执行该函数。他的参数和 call()相同。

10. 显示原型和隐式原型，手绘原型链，原型链是什么？为什么要有原型链？

在 JS 中万物皆对象，方法(FUNCTION)是对象，方法的原型（FUNCTION.PROTOTYPE）是对象，对象具有属性（__PROTO__）称为隐式原型，对象的隐式原型指向构造该对象的构造函数的显式原型。原型对象只存在于函数对象。也就是本质上只要是通过 NEW FUNCTION 创建的函数对象会有一个原型对象

五条原型原则

- | | |
|---|---|
| 1 | 所有的引用类型（数组，对象，函数）都具有对象特性，即可自由扩展属性（除了 null 以外） |
| 2 | 所有的引用类型（数组，对象，函数），都有一个_proto_属性，属性值是一个普通的对象（隐式原型） |
| 3 | 当试图得到一个对象的某个属性时，如果这个对象本身没有这个属性，那么会去它的_proto_(即它的构造函数的 prototype)中寻找 |

原型链解释说明

- | |
|---|
| 1. INSTANCEOF 用于判断引用类型属于那个构造函数的方法，F INSTANCEOF FOO 的判断逻辑是：F 的_proto_一层一层往上，能否对应到 FOO.PROTOTYPE 再试着判断 F INSTANCEOF OBJECT。 |
| 2. INSTANCEOF 用于判断引用类型属于那个构造函数的方法 F INSTANCEOF FOO 的判断逻辑是：F 的_proto_一层一层往上，能否对应到 FOO.PROTOTYPE 再试着判断 F INSTANCEOF OBJECT。 |

11. 你知道哪几种创建对象的方式？

- | |
|-----------------|
| 1 工厂模式 |
| 2 构造函数模式 |
| 3 原型模式 |
| 4 组合使用构造模式和原型模式 |

12. 实现继承有哪些方式，他们的优缺点是什么？

原型继承	优点：父类的方法（getName）得到了复用
	缺点：同理父类的属性（name）也是复用，即子类实例没有自己的属性

构造函数继承	优点：子类的每个实例都有自己的属性（name），不会相互影响
	缺点：但是继承父类方法的时候就不需要这种特性，没有实现父类方法的复用
组合式继承	优点：继承了上述两种方式的优点，摒弃了缺点，复用了方法，子类又有各自的属性
	缺点： 因为父类构造函数被执行了两次，子类的原型对象(Sub.prototype)中也有一份父类的实例属性，而且这些属性会被子类实例(sub1,sub2)的属性覆盖掉，也存在内存浪费
寄生组合式继承	优点：以上优点均有。
	缺点：就是在继承父类方法的时候调用了父类构造函数，从而造成内存浪费，// 现在只要解决了这个问题就完美了。那在复用父类方法的时候，// 使用 Object.create 方法也可以达到目的，没有调用父类构造函数，问题解决
es6 中的 class	

13. New 一个对象具体做了什么 ？

使用关键字 new 创建新实例对象过了以下几步

1	创建一个新对象，如：var person = {}
2	新对象的_proto_属性指向构造函数的原型对象
3	将构造函数的作用域赋值给新对象。（也所以 this 对象指向新对象）
4	执行构造函数内部的代码，将属性添加给 person 中的 this 对象
5	返回新对象 person

14. 请写出 AJAX，或者说出流程 ？

什么是 ajax ：就是异步的 javascript 和 XML

同步是指：阻塞式，必须等待上一个程序执行完后才能继续执行。

异步是指：非阻塞式，就是程序可以同时执行。

你可以扫描二维码查看详情



15. 变量提升如何理解 ？

JavaScript 中，函数及变量的声明都将被提升到函数的最顶部。

JavaScript 中，变量可以在使用后声明，也就是变量可以先使用再声明。

JavaScript 只有声明的变量会提升，初始化的不会

16. 举例说明一个匿名函数的典型用例 ？

```
1  (function () {
2      alert('water');
3  })();
4  // 带参数的也可以
5      (function (o) {
6          alert(o);
7      })('water');
8  
```

17. 指出 JS 的宿主对象和原生对象的区别 ？为什么扩张 JS 内置对象是不提倡的做法？有哪些内置对象和函数举几个例子吧。

内置（Build-in）对象与原生（Naitve）对象的区别在于：前者总是在引擎初始化阶段，就被创建好的对象，是后者的一个子集；而后者包括了一些在运行过程中动态创建的对象。宿主对象不是引擎的原生对象，而是由宿主框架通过某种机制注册到 JavaScript 引擎中的对象。

因为你不知道哪一天浏览器或 javascript 本身就会实现这个方法，而且和你扩展的实现有不一致的表现。到时候你的 javascript 代码可能已经在无数个页面中一直执行，而浏览器的实现导致所有使用扩展原型的代码都崩溃了

String 对象	字符串对象，提供了对字符串进行操作的属性和方法
Array 对象	数组对象，提供了数组操作方面的属性和方法
Date 对象	日期时间对象，可以获取系统的日期时间信息
Boolean 对象	布尔对象，一个布尔变量就是一个布尔对象。(没有可用的属性和方法)
Number 对象	数值对象。一个数值变量就是一个数值对象
Math 对象	数学对象，提供了数学运算方面的属性和方法

18. Attribute 和 property 之间的区别是什么？

property 是 DOM 中的属性，是 JavaScript 里的对象

attribute 是 HTML 标签上的特性，它的值只能是字符串

19. Document load 和 document DOMContentLoaded 两个事件之前的区别？

说明：他们的区别是，触发的时机不一样，先触发 DOMContentLoaded 事件，后触发 load 事件

DOM 文档加载的步骤为

1	解析 HTML 结构
2	加载外部脚本和样式表文件
3	解析并执行脚本代码
4	DOM 树构建完成。//DOMContentLoaded
5	加载图片等外部文件
6	页面加载完毕。//load

备注：在第 4 步，会触发 DOMContentLoaded 事件。在第 6 步，触发 load 事件

20. === 和 == , [] === [], undefined === undefined,[] == [], undefined == undefined

===: 三个等号我们称为等同符,当等号两边的值为相同类型的时候,直接比较等号两边的值,值相同则返回 true,若等号两边的值类型不同时直接返回 false

==: 两个等号我们称为等值符,当等号两边的值为相同类型时比较值是否相同,类型不同时会发生类型的自动转换,转换为相同的类型后再作比较

21. Typeof 能够得到哪些值？

number	boolean	string	undefined	object	function
--------	---------	--------	-----------	--------	----------

22. 什么是 “use strict”,好处和坏处是什么？

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：

- 1：消除 Javascript 语法的一些不合理、不严谨之处，减少一些怪异行为；
- 2：消除代码运行的一些不安全之处，保证代码运行的安全；
- 3：提高编译器效率，增加运行速度；
- 4：为未来新版本的 Javascript 做好铺垫。

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

23. 函数的作用域是什么？JS 的作用域你了解吗？

函数作用域的含义是指，属于这个函数的全部变量都可以在整个函数的范围内使用及复用(事实上在嵌套的作用域中也可以使用)。这种设计方案是非常有用的，能充分利用 JavaScript 变量可以根据需要改变值类型的“动态”特性

1.全局变量：声明在函数外部的变量（所有没有 var 直接赋值的变量都属于全局变量）

2.局部变量：声明在函数内部的变量（所有没有 var 直接赋值的变量都属于全局变量）

全局变量在整个上下文都有效只是在没有赋值之前调用，会输出 undefined

函数作用域:是针对局部变量来说的，在函数中定义的变量在函数外不能获取

块级作用域:概念 “{ }” 中间的部分都是块级作用域 ex: for while if ，js 中没有块级作用域，但是可以用闭包实现类似功能

24. Js 如何实现重载和多态？

多态： 多态的实现可以采用和继承类似的方法。首先定义一个抽象类，其中调用一些虚方法，虚方法在抽象类中没用定义，而是通过其具体的实现类来实现

重载：	重载的定义是指函数的方法名相同，但参数不同
------------	-----------------------

JS 中如何实现方法重载？这涉及到三个问题

1	同名函数的调用问题
2	函数中特殊的参数 arguments
3	如何利用 arguments 实现方法重载

实现重载主要就是利用 ARGUMENTS

25. 常用的数组 API，字符串 API 有哪些？

常用 API	
操作方法如下	
concat():	基于当前数组，创建一个新的数组，并返回这个新数组，不会改变原数组
slice():	可以接受一个或两个参数，要返回项的起始和结束位置，返回所截取数组的项，但是不包括结束位置的项，不会改变原数组
splice():	主要用法是向数组的中部插入项，会改变原数组，是最强大的数组方法。最多指定 3 个参数
indexOf()和 lastIndexOf()。这两个方法都接受两个参数：要查找的项和表示查找起点位置的索引。indexOf()从数组的开头开始向后查找，lastIndexOf()从数组的末尾开始向前查找	
迭代方法	
A	every():对数组的每一项运行给定函数，如果数组的每一项都返回 true，则返回 true。对数组应用该方法，有返回值为 true 或 false
B	some():对数组的每一项运行给定函数，如果数组的任一项都返回 true，则返回 true。对数组应用该方法，有返回值为 true 或 false

C	filter():对数组的每一项运行给定函数，返回该函数中会返回 true 的项组成的数组。有返回值，为符合条件的项组成的数组
D	map():对数组的每一项运行给定函数,有返回每次函数调用的结果组成的数组
E	forEach():对数组的每一项运行给定函数,这个方法没有返回值。本质上与使用 for 循环迭代数组是一样的
归并方法	
<p>reduce()和 reduceRight()。他们的区别在于从哪头开始遍历数组，除此之外，它们完全相同。这两个方法都接收两个参宿：一个在每一项上调用的函数和(可选)作为归并基础的初始值。传给 reduce()和 reduceRight()的函数接收 4 个参数：前一个值、当前值、项的索引和数组对象</p>	

26. 原生事件绑定（跨浏览器），下面案例中 dom0 与 dom2 的区别是？

Dom0
<p>通过 javascript 制定事件处理程序的传统方式。就是将一个函数赋值给一个事件处理属性。第四代 web 浏览器出现，至今为所有浏览器所支持。优点，简单且具有跨浏览器的优势。</p>
Dom2
<p>DOM2 级事件处理方式指定了，添加事件处理程序和删除事件处理程序的方法。</p>
<p>区别：</p> <p>dom0 级后面绑定的事件会覆盖前面绑定的事件，点击的时候会执行新绑定的</p> <p>dom0 一个事件处理程序只能对应一个处理函数</p> <p>dom2 级不会覆盖前面绑定的事件，点击的时候所有绑定的事件会依次执行一遍</p>

27. 给定一个元素获取它相对于视图窗口的坐标？

1	clientHeight 和 clientWidth 用于描述元素内尺寸，是指 元素内容+内边距 大小，不包括边框（IE 下实际包括）、外边距、滚动条部分
2	offsetHeight 和 offsetWidth 用于描述元素外尺寸，是指 元素内容+内边距+边框，不包括外边距和滚动条部分

3	clientTop 和 clientLeft 返回内边距的边缘和边框的外边缘之间的水平和垂直距离，也就是左，上边框宽度
4	offsetTop 和 offsetLeft 表示该元素的左上角（边框外边缘）与已定位的父容器（offsetParent 对象）左上角的距离
5	offsetParent 对象是指元素最近的定位（relative,absolute）祖先元素，递归上溯，如果没有祖先元素是定位的话，会返回 null 可以通过调用元素的 getBoundingClientRect 方法。方法返回一个有 left、right、top、bottom 属性的对象，分别表示元素四个位置的相对于视口的坐标。getBoundingClientRect 所返回的坐标包含元素的内边距和边框，不包含外边距

28. Js 的字符串类型有哪些方法？正则表达式的函数怎么使用？

1	charCodeAt 方法返回一个整数，代表指定位置字符的 Unicode 编码
2	fromCharCode 方法从一些 Unicode 字符串中返回一个字符串
3	charAt 方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串
4	slice 方法返回字符串的片段
5	substring 方法返回位于 String 对象中指定位置的子字符串
6	substr 方法返回一个从指定位置开始的指定长度的子字符串
7	indexOf 方法放回 String 对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1
8	lastIndexOf 方法返回 String 对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1
9	search 方法返回与正则表达式查找内容匹配的第一个字符串的位置
10	concat 方法返回字符串值，该值包含了两个或多个提供的字符串的连接
11	将一个字符串分割为子字符串，然后将结果作为字符串数组返回
12	toLowerCase 方法返回一个字符串，该字符串中的字母被转换成小写
13	toUpperCase 方法返回一个字符串，该字符串中的所有字母都被转换为大写字母

js 正则表达式之 replace 函数用法：

功能： replace 函数返回根据正则表达式进行文字替换后的字符串的复制

格式： stringObj.replace(rgExp, replaceText)

参数： 字符串 stringObj, rgExp 正则表达式, replaceText 所替换的内容

js 正则表达式之 test 函数用法

功能介绍： 该方法的返回值是布尔值，通过该值可以匹配字符串中是否存在着正则表达式相匹配的结果，如果有匹配内容，返回 true，如果没有匹配内容返回 false，该方法常用于判断用户输入数据的合法性，比如检验 Email 的合法性

js 正则表达式之 match 函数用法

功能： 使用正则表达式模式对字符串执行查找，并将包含查找的结果作为数组返回

格式： stringObj.match(rgExp) stringObj 为字符串必选 rgExp 为正则表达式必选项

返回值： 如果能匹配则返回结果数组，如果不能匹配返回 null

js 正则表达式之 search 方法讲解

功能： 返回与正则表达式查找内容匹配的字符串的第一个子字符串的位置

语法： stringObj.search(rgExp) stringObj 必选项 rgExp 正则表达式

返回值： search 方法指明是否存在相应的匹配。如果找到一个匹配，search 方法将返回一个整数值，指明这个匹配距离字符串开始的偏移位置。如果没有找到匹配，则返回 -1

js 正则表达式之 exec 方法讲解

功能说明： 该函数通过对指定你的字符串进行一次匹配检测，获取字符串中的第一个与正则表达式的内容，并且将匹配的内容和子匹配的结果存放在返回数组中。

基本方法： objReg.exec(string) , objReg, RegExp 对象的名称 , string, 要进行匹配的字符串 。

29. 深拷贝原理是什么，请简要描述过程？

浅拷贝不能完成需求，对于属性是对象的时候，只是进行简单的地址拷贝，其引用关系也在。

```
//深拷贝 两者之间改变互不影响
//1 拷贝后两者之间不再存在共享关系
//2 拷贝之后数据类型不能发生改变，也就是需要判断是数组的时候，需要进行单独递归的遍历
//3 在继承的时候，我们通过原型属性实现原型对象属性的继承，在进行深拷贝的时候，我们首先需要提出原型对象上的属性；通过hasOwnProperty方法来进行筛选；
```

深拷贝和浅拷贝的区别：是在对象状态中包含其它对象的引用的时候，当拷贝一个对象时，如果需要拷贝这个对象引用的对象，则是深拷贝，否则是浅拷贝。

具体案例的分析你可以扫一扫

30. 你能编写一个可复用的事件监听方法吗？

当同一个对象使用.onclick 的写法触发多个方法的时候，后一个方法会把前一个方法覆盖掉，也就是说，在对象的 onclick 事件发生时，只会执行最后绑定的方法。而用事件监听则不会有覆盖的现象，每个绑定的事件都会被执行。

如下：

原生态的事件绑定函数addEventListener：

```
var eventOne = function(){
    alert("第一个监听事件");
}
function eventTwo(){
    alert("第二个监听事件");
}
window.onload = function(){
    var btn = document.getElementById("yuanEvent");
    //addEventListener: 绑定函数
    btn.addEventListener("click",eventOne);
    btn.addEventListener("click",eventTwo);
}
//输出：第一个监听事件 和 第二个监听事件
```

原生态的事件绑定函数 addEventListener

```
var eventOne = function(){
    alert("第一个监听事件");
}
function eventTwo(){
    alert("第二个监听事件");
}
window.onload = function(){
    var btn = document.getElementById("yuanEvent");
    //addEventListener: 绑定函数
    btn.addEventListener("click",eventOne);
    btn.addEventListener("click",eventTwo);
}
//输出：第一个监听事件 和 第二个监听事件
```

//2、采用事件监听给对象绑定方法后，可以解除相应的绑定，写法如下：

```
var eventOne = function(){
    alert("第一个监听事件");
}
function eventTwo(){
    alert("第二个监听事件");
}
window.onload = function(){
    var btn = document.getElementById("yuanEvent");
    btn.addEventListener("click",eventOne);
    btn.addEventListener("click",eventTwo);
    btn.removeEventListener("click",eventOne);
} //输出：第二个监听事件
```

31. 浏览器（web 端）cookie 如何设置或获取？

```
<script>
    //设置cookie
    function setCookie(cname, cvalue, exdays) {
        var d = new Date();
        d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
        var expires = "expires=" + d.toGMTString();
        document.cookie = cname + "=" + cvalue + ";" + expires;
    } //获取cookie
    function getCookie(cname) {
        var name = cname + "=";
        var ca = document.cookie.split('; ');
        for (var i = 0; i < ca.length; i++) {
            var c = ca[i].trim();
            if (c.indexOf(name) == 0) return c.substring(name.length, c.length);
        }
        return "";
    }
</script>
```

32. setTimeout 和 promise 的执行顺序是什么？

setTimeout 有一个 4ms 的最短时间，也就是说不管你设定多少，反正最少都要间隔 4ms 才运行里面的回调，从具体实现上来说，这俩的异步队列不一样，Promise 所在的那个异步队列优先级要高一些

33. JavaScript 的事件流模型都有什么？

捕获流： 在捕获模式的定义中，当事件发生时，该事件由高层次向低层次传递

冒泡流： 在冒泡流的定义中，当事件发生时，该事件由低层次向高层次传播（像是水泡向上冒出）

DOM 中的事件流： 当事件发生时，事件由捕获过程->冒泡过程，即由捕获阶段->处于目标对象阶段->冒泡

34. navigator 对象, location 和 history 你了解多少?

Navigator 对象

- 1 History 对象包含用户（在浏览器窗口中）访问过的 URL。
- 2 History 对象是 window 对象的一部分，可通过 window.history 属性对其进行访问。

Navigator 对象方法

- 1 Back():加载 history 列表中的前一个 URL
- 2 Forward():加载 history 列表中的下一个 URL
- 3 Go():加载 history 列表中的某个具体页面。

Location 对象

- 1 Location 对象包含有关当前 URL 的信息。
- 2 Location 对象是 Window 对象的一个部分，可通过 window.location 属性来访问。

Location 对象方法

hash 设置或返回从井号 (#) 开始的 URL（锚）。

host 设置或返回主机名和当前 URL 的端口号。

hostname 设置或返回当前 URL 的主机名。

href 设置或返回完整的 URL。

pathname 设置或返回当前 URL 的路径部分。

port 设置或返回当前 URL 的端口号。

protocol 设置或返回当前 URL 的协议。

search 设置或返回从问号 (?) 开始的 URL（查询部分）。

assign() 加载新的文档。

reload() 重新加载当前文档。

replace() 用新的文档替换当前文档。

35. js 的垃圾回收机制（了解）？

javascript 具有自动垃圾回收机制，执行环境会负责管理代码执行过程中使用的内存。原理就是找出那些不再继续使用的变量，然后释放其占有内存

36.说一下内存泄漏的原因和场景

程序的运行需要内存。只要程序提出要求，操作系统或者运行时（runtime）就必须供给内存。

对于持续运行的服务进程（daemon），必须及时释放不再用到的内存。否则，内存占用越来越高，轻则影响系统性能，重则导致进程崩溃。

不再用到的内存，没有及时释放，就叫做内存泄漏（memory leak）。

具体的使用方法你可以参考



36. Dom 事件中的 target 和 currentTarget 有什么区别？

target 在事件流的目标阶段；currentTarget 在事件流的捕获，目标及冒泡阶段。只有当事件流处在目标阶段的时候，两个的指向才是一样的，而当处于捕获和冒泡阶段的时候，target 指向被单击的对象而 currentTarget 指向当前事件活动的对象(注册该事件的对象)（一般为父级）

37. Typeof 和 instanceof 区别，instanceof 的原理是什么？

Typeof:	typeof 是一个一元运算，放在一个运算数之前，运算数可以是任意类型。 它返回值是一个字符串，该字符串说明运算数的类型
instanceof	用于判断一个变量是否某个对象的实例
原理： nstanceof 检测一个对象 A 是不是另一个对象 B 的实例的原理是：查看对象 B 的 prototype 指向的对象是否在对象 A 的[[prototype]]链上。如果在，则返回 true,如果不在则返回 false。不过有一个特殊的情况，当对象 B 的 prototype 为 null 将会报错(类似于空指针异常)	

38. JS 动画和 CSS3 动画有什么区别？

功能涵盖面，JS 比 CSS3 大定义动画过程的@keyframes 不支持递归定义，如果有多种类似的动画过程，需要调节多个参数来生成的话，将会有很大的冗余（比如 jQuery Mobile 的动画方案），而 JS 则天然可以以一套函数实现多个不同的动画过程时间尺度上，@keyframes 的动画粒度粗，而 JS 的动画粒度控制可以很细

CSS3 动画里被支持的时间函数非常少，不够灵活以现有的接口，CSS3 动画无法做到支持两个以上的状态转化实现/重构难度不一，CSS3 比 JS 更简单，性能调优方向固定对于帧速表现不好的低版本浏览器，CSS3 可以做到自然降级，而 JS 则需要撰写额外代码 CSS 动画有天然事件支持（TransitionEnd、AnimationEnd，但是它们都需要针对浏览器加前缀），JS 则需要自己写事件 CSS3 有兼容性问题，而 JS 大多时候没有兼容性问题

39. JS 实现一个简单的倒计时程序？

```
<script>
  var remaining_time = document.getElementById('remaining_time');
  var t = 6;
  function changeTime() {
    t--;
    remaining_time.innerHTML = "&nbsp;" + t + "&nbsp;";
    if (t == 0) {
      window.location.href = 'http://www.jquerycn.cn';
      return;
    }
    setTimeout('changeTime()', 1000);
  }
  changeTime();
</script>
```

40. JS 处理异常的语句有哪些举例说明？

Javascript 的异常捕获机制	
1	try...catch 语句
try{ console.log(b); console.log("我不会输出的，不要找了") }catch(error){ console.log("发生错误了") } console.log("我 try catch 后面的代码")	
2	finally 语句


```
try{ console.log(b); console.log("我不会输出的,不要找了") }catch(error){ console.log("发生错误了") }finally  
  
{ console.log("不管发生不发生错误,我都会执行") } console.log("我 try catch 后面的代码")
```

41. JS 的设计模式你知道哪些？

1	工厂模式
优点	能解决多个相似的问题
缺点	不能知道对象识别的问题(对象的类型不知道)
2	单体模式
单体模式是一个用来划分命名空间并将一批属性和方法组织在一起的对象，如果它可以被实例化，那么它只能被实例化一次	
单体模式的优点是：	
1	可以用来划分命名空间，减少全局变量的数量
2	使用单体模式可以使代码组织的更为一致，使代码容易阅读和维护
3	可以被实例化，且实例化一次
3	模块模式
模块模式使用了一个返回对象的匿名函数。在这个匿名函数内部，先定义了私有变量和函数，供内部函数使用，然后将一个对象字面量作为函数的值返回，返回的对象字面量中只包含可以公开的属性和方法。这样的话，可以提供外部使用该方法；由于该返回对象中的公有方法是在匿名函数内部定义的，因此它可以访问内部的私有变量和函数	
我们什么时候使用模块模式？	
如果我们必须创建一个对象并以某些数据进行初始化，同时还要公开一些能够访问这些私有数据的方法，那么我们这个时候就可以使用模块模式了	
4	代理模式
代理的优点：	

1	代理对象可以代替本体被实例化，并使其可以被远程访问；
2	它还可以把本体实例化推迟到真正需要的时候；对于实例化比较费时的本体对象，或者因为尺寸比较大以至于不用时不适于保存在内存中的本体，我们可以推迟实例化该对象

42. 轮播图组件开发，我有 10000 万张图片要轮播？

首先让一组图片绝对定位，让其重叠在一起，然后通过一个函数来控制图片的透明度变化。同时还有一个定时器，不停的除发这个函数，每次改变不同图片的透明度，让其显示。（更简单的效果是直接修改 display 属性，让该显示的图片 display:block,而不显示的设为 display:none 就可以，只是效果上会差一些，但原理相同）

1	图片切换函数。接受一个参数，表示滚动方向。调用缓动函数切换图片。调用切换按钮图标函数点亮相应的按钮
2	缓动函数
3	点亮按钮函数
4	初始化函数。用于绑定事件，创建按钮和箭头，初始化最初位置
5	创建箭头函数
6	创建按钮函数
7	开始轮播函数
8	轮播函数
9	停止函数。用于停止轮播

还有一些公用方法：`$()`：选择 DOM 元素。`addClass(ele,"className")`：给元素添加类名。
`removeClass(ele,"className")`移除元素的类名。`$.add(ele,"type",fun)`：给一个 DOM 节点绑定事件。
`getCSS(ele,"prop")`：获取元素相应属性的值。`$.delegateTag("selector","tagName","type",fun)`：事件代理。

43. 说一说 websocket 的工作原理和机制？

WebSocket 是一种双向通信协议，在建立连接后，WebSocket 服务器和 Browser/Client Agent 都能主动的向对方

发送或接收数据，就像 Socket 一样；WebSocket 需要类似 TCP 的客户端和服务端通过握手连接，连接成功后才能相互通信

44. 手指点击触控屏幕是什么事件，函数柯里化是什么？

手指点击称为：touch 事件；

什么是函数柯里化：

柯里化是一个转换过程，把接受多个参数的函数变换成接受一个单一参数(译注：最初函数的第一个参数)的函数，如果其他的参数是必要的，返回接受余下的参数且返回结果的新函数

柯理化函数思想：一个 js 预先处理的思想；利用函数执行可以形成一个不销毁的作用域的原理，把需要预先处理的内容都储存在这个不销毁的作用域中，并且返回一个小函数，以后我们执行的都是小函数，在小函数中把之前预先存储的值进行相关的操作处理即可；

bind 方法的作用：把传递进来的 callback 回调方法中的 this 预先处理为上下文 context;

45. JS 代码调试？

1

可以在 JavaScript 代码中加入一句 debugger;来手工造成一个断点效果。

需要带有条件的断点吗？你只需要用 if 语句包围它

2

设置在 DOM node 发生变化时触发断点，谷歌浏览器的开发工具里有一个超级好用的功能，专门可以对付这种情况，叫做“Break on...”，你在 DOM 节点上右键，就能看到这个菜单项。断点的触发条件可以设置成这个节点被删除、节点的属性有任何变化，或它的某个子节点有变化发生

3

Ajax 断点：XHR 断点，或 Ajax 断点，就像它们的名字一样，可以让我们设置一个断点，在特点的 Ajax 调用发生时触发它们。

当你在调试 Web 应用的网络传输时，这一招非常的有效

4

移动设备模拟环境：谷歌浏览器里有一些非常有趣的模拟移动设备的工具，帮助我们调试程序在移动设备里的运行情况。找到它的方法是：按 F12，调出开发者工具，然后按 ESC 键(当前 tab 不能是

	Console), 你就会看到第二层调试窗口出现, 里面的 Emulation 标签页里有各种模拟设备可选。当然, 这不会就变成了真正的 iPhone, 只是模拟了 iPhone 的尺寸, 触摸事件和浏览器 User Agent 值
5	使用 Audits 改进你的网站 YSlow 是一个非常棒的工具。谷歌浏览器的开发者工具里也有一个非常类似的工具, 叫 Audits。 它可快速的审计你的网站, 给你提出非常实际有效的优化你的网站的建议和方法

4.ES6

1. 聊聊 promise?

promise 是一个异步函数,主要是为了解决异步处理回调地狱(也就是循环嵌套的问题)而产生的,有 3 种状态,Fulfilled 为成功的状态,Rejected 为失败的状态,Pending 既不是 Fulfilled 也不是 Rejected 的状态,可以理解为 Promise 对象实例创建时候的初始状态,要会写 promise 这个方法

2. ES6 特性你了解多少? 如果遇到一个东西不知道是 ES6 还是 ES5, 你该如何区分?

常用的 es6 新特性:

1. let && const

let 命令也用于声明对象,但是作用域为局部。

2. iterable 类型

为了统一集合类型,ES6 标准引入了新的 iterable 类型,Array、Map 和 Set 都属于 iterable 类型,具有 iterable 类型的集合可以通过新的 for ... of 循环来遍历。

3. 解构赋值

4. 箭头函数

5. 类



es6 比 es5 增加了很多特殊的方法，如果你遇到了这些特殊的方法，你就可以确定它是 es6。

如果你的代码中没有引用这些特殊的方法，那我们就可以认为他是 es5 的。

所以前提你需要了解 es6 的语法才能做判断，高频使用的特性有箭头函数、解构赋值、let、const。

3. ES6 的继承和 ES5 的继承有什么区别？

es5 的继承是通过原型或者是构造函数机制来实现，es6 用过 class 关键字定义类，里面有构造方法，类之间通过 extends 关键字实现，子类必须在 constructor 方法中调用 super 方法

4. Promise 如何封装一个 AJAX？

```
function ajax(optionsOverride) {
  // 将传入的参数与默认设置合并
  var options = {};
  for (var k in ajaxOptions) {
    options[k] = optionsOverride[k] || ajaxOptions[k];
  }
  options.async = options.async === false ? false : true;
  var xhr = options.xhr = options.xhr || new XMLHttpRequest();

  return new Promise(function (resolve, reject) {
    xhr.open(options.method, options.url, options.async);
    xhr.timeout = options.timeout;

    //设置请求头
    for (var k in options.headers) {
      xhr.setRequestHeader(k, options.headers[k]);
    }
    // 注册xhr对象事件
    xhr.onprogress = options.onprogress;
    xhr.upload.onprogress = options.onuploadprogress;
    xhr.responseType = options.dataType;

    xhr.onabort = function () {
      reject(new Error({
        errorType: 'abort_error',
        xhr: xhr
      }));
    };
    xhr.ontimeout = function () {
      reject({
        errorType: 'timeout_error',
        xhr: xhr
      });
    };
    xhr.onerror = function () {
      reject({
        errorType: 'onerror',
        xhr: xhr
      });
    };
    xhr.onloadend = function () {
      if ((xhr.status >= 200 && xhr.status < 300) || xhr.status === 304)
        resolve(xhr);
      else
        reject({
          errorType: 'status_error',
          xhr: xhr
        });
    };
  });
}
```

```

    })
  }
  try {
    xhr.send(options.data);
  }
  catch (e) {
    reject({
      errorType: 'send_error',
      error: e
    });
  }
})
</script>

```

5. let 和 const 的优点？

let 和 const 有了块级作用域，变量声明不会提升相比于 var

6. ES6 generator 是什么，async / await 实现原理？

generator 函数就是一个封装的异步任务，也就是异步任务的容器，执行 Generator 函数会返回一个遍历器对象,async 函数的实现，就是将 Generator 函数和自动执行器，包装在一个函数里

7. ES6 和 node 的 commonjs 模块化规范的区别。

es6 是 js 的增强版，是 js 的语法规则，commonjs 都只是为了解决 js 文件之间的依赖和引用问题，所以是一种 js 的包管理规范，其中的代表是 Node 遵循 commonjs 规范

8. 箭头函数，以及他们的 this。

所谓的箭头函数是在写法上面很简便和之前相比，类似于:(a,b)=>{return a+b;}，箭头函数的 this 默认指向在定义它时,它所处的对象,而不是执行时的对象，定义它的时候,可能环境是 window（即继承父级的 this）

5.计算机网络相关

1. HTTP 协议头包含哪些重要部分，HTTP 状态码。

HTTP 协议头包括通用头，请求头，响应头和实体头四个部分

200	请求已成功，请求所希望的响应头或数据体将随此响应返回
-----	----------------------------

302	Moved Permanently（重定向）请求的 URL 已移走。Response 中应该包含一个 Location URL，说明资源现在所处的位置
304	Not Modified（未修改）客户的缓存资源是最新的，要客户端使用缓存
404	Not Found 未找到资源，可能是路径方面的错误
503	服务器目前无法为请求提供服务，但过一段时间就可以恢复服务

2. 网络 url 输入到输出都做了什么？。

1	当我们输入一个域名像：www.baidu.com
2	浏览器查找浏览器缓存，如果有域名的 IP 地址则返回，如果没有继续查找
3	系统查找系统缓存，如果有域名的 IP 地址则返回，如果没有继续查找
4	路由器查找路由器缓存，如果有域名的 IP 地址则返回，如果没有继续查找
5	本地域名服务器采用迭代查询，它先向一个根域名服务器查询
6	根域名服务器告诉本地域名服务器，下一次应查询的顶级域名服务器 dns.com 的 IP 地址
7	本地域名服务器向顶级域名服务器 dns.com 进行查询
8	顶级域名服务器 dns.com 告诉本地域名服务器，下一次应查询的权限域名服务器 dns.baidu.com 的 IP 地址
9	本地域名服务器向权限域名服务器 dns.baidu.com 进行查询
10	权限域名服务器 dns.baidu.com 告诉本地域名服务器，所查询的主机 www.baidu.com 的 IP 地址
11	本地域名服务器最后把查询结果告诉主机
12	主机浏览器获取到 Web 服务器的 IP 地址后，与服务器建立 TCP 连接
13	浏览器所在的客户机向服务器发出连接请求报文
14	服务器接收报文后，同意建立连接，向客户机发出确认报文
15	客户机接收到确认报文后，再次向服务器发出报文，确认已接收到确认报文
16	此处客户机与服务器之间的 TCP 连接建立完成，开始通信

17	浏览器发出取文件命令：GET
18	服务器给出响应，将指定文件发送给浏览器
19	浏览器释放 TCP 连接
20	浏览器所在主机向服务器发出连接释放报文，然后停止发送数据
21	服务器接收到释放报文后发出确认报文，然后将服务器上未传送完的数据发送完
22	服务器数据传输完毕后，向客户机发送连接释放报文
23	客户机接收到报文后，发出确认，然后等待一段时间后，释放 TCP 连接
24	浏览器显示页面中所有文本

3. 为什么说性能优化就要减少 HTTP 的访问次数？。

1	http 请求头的数据量：每次请求都会带上一些额外的信息进行传输,所以请求越多的时候，在网络上传输的数据就会变多了，速度就变慢了
2	http 连接的开销：从用户输入 URL 到看到页面，经过一系列的解析和连接，已经等待

4. 描述一下 HTTP 的请求过程与原理？。

http 请求的过程：域名解析 --> 发起 TCP 的 3 次握手 --> 建立 TCP 连接后发起 http 请求 -->服务器响应 http 请求，浏览器得到 html 代码 -->浏览器解析 html 代码，并请求 html 代码中的资源（如 js、css、图片等） --> 浏览器对页面进行渲染呈现给用户

请求原理：HTTP 协议是应用层的一种协议，是一种 C/S 架构服务，基于 TCP/IP 协议来通信，监听在 TCP 的 80 端口上，HTTP 协议实现的是客户端可以向服务端获得 web 资源

5. https 有几次握手和挥手？https 的原理什么是？。

https 是 3 次握手和 4 次挥手，和 http 是一样的。

原理：https 在传输数据前需要客户端(浏览器)与服务器(网站)之间进行一次握手,在握手过程中将确立双方

加密传输数据的密码信息.TLS/SSL 协议是一套加密传输协议,使用了非对称加密,对称加密,以及 HASH 算法

6. http 有几次握手和挥手? TLS 的中文名? TLS 在那一网络层? 。

http 是 3 次握手和 4 次挥手, TLS 的中文名是: 安全传输层协议, 在传输层。

7. TCP 连接的特点, TCP 连接如何保证安全可靠?

TCP 的可靠性是通过顺序编号和确认 (ACK) 来实现的, TCP 的接收端必须丢弃重复的数据, 并且 TCP 提供流量控制, 连接的每一个地方都有固定大小的缓冲空间。

8. 为什么 TCP 连接需要三次握手? 。

三次握手为了确认客户端跟服务器都能接受到对方的信息,两次的话服务器不能确认客户端能否接收自己发的包

第一次握手, 客户端给服务器发包。此时服务器确认自己可以接收客户端的包, 客户端不确认服务器是否接收到了自己发的包

第二次握手, 服务器端回复客户端。此时客户端确认自己发的包被服务器收到, 也确认自己可以正常接收服务器包, 客户端对此次通信没有疑问了。服务器可以确认自己能接收到客户端的包, 但不能确认客户端能否接收自己发的包

第三次握手, 客户端回复服务器。客户端已经没有疑问了, 服务器也确认刚刚客户端收到了自己的包。两边都没有问题, 开始通信

9. 为什么 TCP 连接需要三次握手四次挥手? 。

为什么是三次握手?

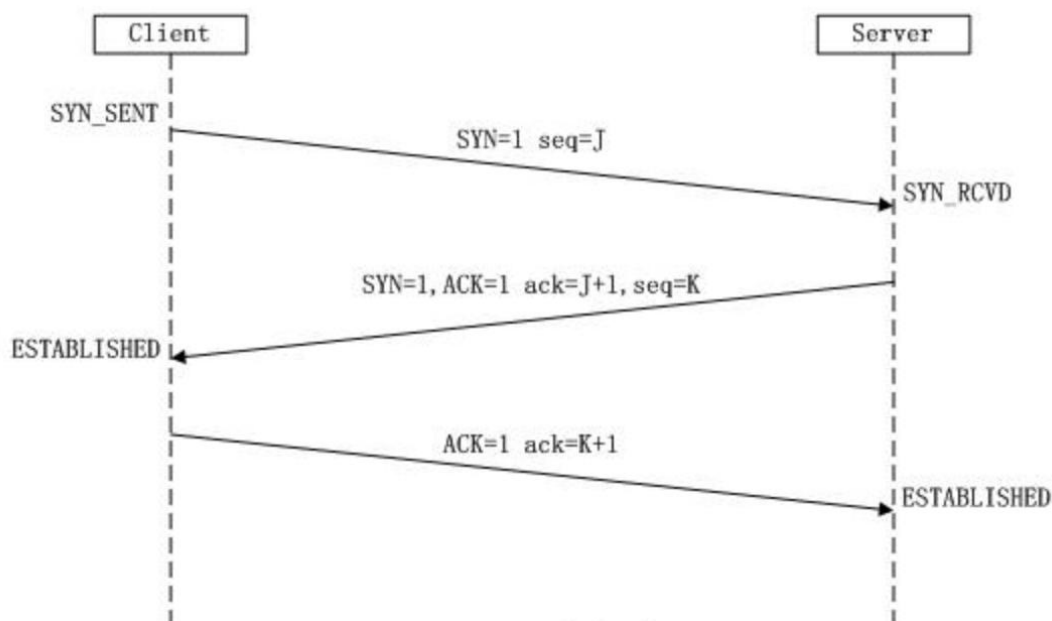
为了防止已失效的连接请求报文段突然有送到了服务器, 因而产生错误,假设两次握手时, 客户发出的第一个请求连接报文段在某一网络节点长时间滞留, 以致延误到连接释放后才到达服务器。服务器收到失效的连接请求报文段后, 认为是客户又发出一次新的连接请求。于是向客户发送确认报文段, 同意建立连接, 此时在假定两次握手的前提下, 连接建立成功。这样会导致服务器的资源白白浪费

为什么是四次挥手？

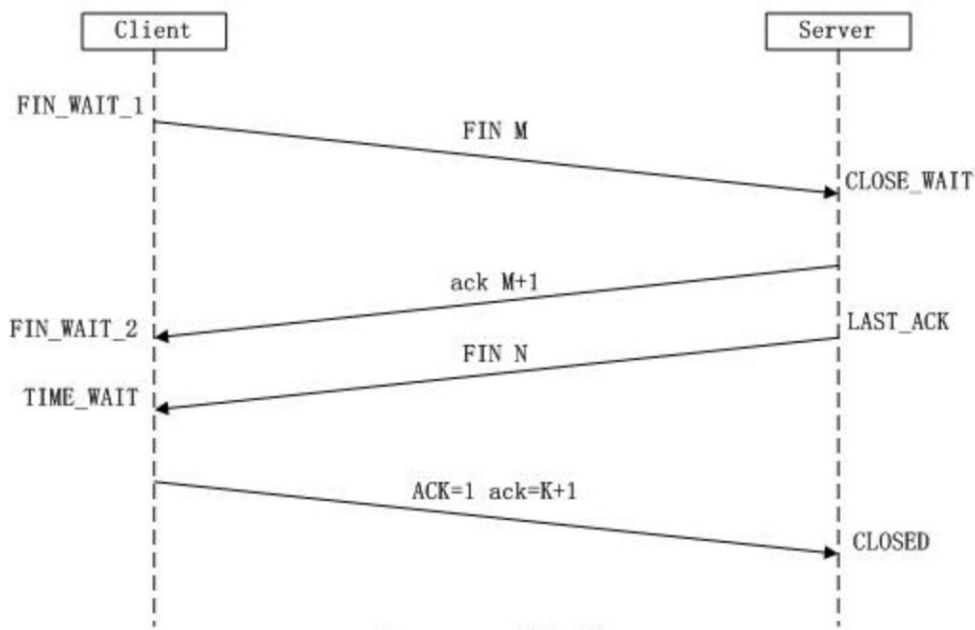
TCP 协议是全双工通信，这意味着客户端和服务端都可以向彼此发送数据，所以关闭连接是双方都需要确认的
共同行为，假设是三次挥手时，首先释放了客户到服务器方向的连接，此时 TCP 连接处于半关闭状态，这时客
户不能向服务器发送数据，而服务器还是可以向客户发送数据。如果此时客户收到了服务器的确认报文段后，就
立即发送一个确认报文段，这会导致服务器向客户还在发送数据时连接就被关闭。这样会导致客户没有完整收到
服务器所发的报文段

10. TCP 的三次握手和四次挥手绘图（当场画写 ACK 和 SEQ 的值）？。

所谓三次握手（Three-Way Handshake）即建立 TCP 连接，就是指建立一个 TCP 连接时，需要客户端和服务端总
共发送 3 个包以确认连接的建立。由客户端执行 `connect()` 触发



所谓四次挥手（Four-Way Wavehand）即终止 TCP 连接，就是指断开一个 TCP 连接时，需要客户端和服务端总共
发送 4 个包以确认连接的断开。由客户端或服务端任一方执行 `close` 来触发



11. TCP 与 UDP 的区别有哪些？

TCP (Transmission Control Protocol 传输控制协议) 是一种面向连接的、可靠的、基于字节流的传输层通信协议

UDP (User Datagram Protocol 用户数据报协议) 是 OSI (Open System Interconnection, 开放式系统互联) 参考模型中一种无连接的传输层协议, 提供面向事务的简单不可靠信息传送服务

TCP 是面向连接的传输控制协议, 而 UDP 提供了无链接的数据报服务//类似电话与短信

TCP 面向连接, 提供可靠的数据服务

TCP 首部开销 20 字节, UDP 首部开销 8 字节

TCP 逻辑通信信道是全双工的可靠信道, UDP 则是不可靠信道

UDP 没有拥塞机制, 因此网络出现拥堵不会使源主机的发送效率降低 (有利于实时会议视频等)

TCP 的连接只能是点到点的, UDP 支持一对一, 多对一, 多对多的交互通信

12. Get 和 Post 的区别? 什么情况下用到?

- 1 GET 使用 URL 或 Cookie 传参。而 POST 将数据放在 BODY 中
- 2 GET 的 URL 会有长度上的限制, 则 POST 的数据则可以非常大
- 3 POST 比 GET 安全, 因为数据在地址栏上不可见

最本质的区别

Get 是用来从服务器上获得数据，而 post 是用来向服务器上传递数据

若符合下列任一情况，则用 post 方法：

- a 请求的结果有持续性的作用，例如：数据库内添加新的数据行
- b 若使用 get 方法，则表单上收集的数据可能让 URL 过长
- c 要传送的数据不是采用 ASCII 编码

若符合下列任一情况，则用 get 方法：

- a 请求是为了查找资源，html 表单数据仅用来搜索
- b 请求结果无持续性的副作用
- c 收集的数据及 html 表单内的输入字段名称的总长不超过 1024 个字符

13. HTTP2 / HTTP1 之间的区别是什么？。

1	http2 采用二进制格式而非文本格式，比起文本格式，二进制格式解析起来更加高效，并且错误少
2	http2 是完全的多路复用，非有序并阻塞的----只需要一个连接即可实现并行，多路复用的意思是它能同时处理多个消息的请求和响应，http1 是一个连接一次只能提交一个请求的效率比较高，多了就会变慢
3	使用报头压缩，http2 降低了开销，http1 的消息头很大冗余，http2 是将消息头中的不同的部分分别用不用的索引进行表示，且会用哈夫曼编码压缩字符串，最后封装成 frame
4	http2 让服务器可以将响应主动”推送”到客户端缓存中，HTTP2 中服务器会主动将资源推送给客户端，例如把 js 和 css 文件主动推送给客户端而不用客户端解析 HTML 后请求再响应

14. 介绍一下 websocket？。

websocket 是一种网络通信协议，是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通信的协议，这个对比着 http 协议来说，http 协议是一种无状态的、无连接的、单向的应用层协议，通信请求只能由客户端发起，服务端对请求

做出应答处理。http 协议无法实现服务器主动向客户端发起消息，WebSocket 连接允许客户端和服务端之间进行全双工通信，以便任一方都可以通过建立的连接将数据推送到另一端。WebSocket 只需要建立一次连接，就可以一直保持连接状态

```
<script>
// 初始化一个 WebSocket 对象
var ws = new WebSocket("ws://localhost:9998/echo");
// 建立 web socket 连接成功触发事件
ws.onopen = function () {
// 使用 send() 方法发送数据
ws.send("发送数据");
alert("数据发送中..."); };
// 接收服务端数据时触发事件
ws.onmessage = function (evt) { var received_msg = evt.data; alert("数据已接收..."); };
// 断开 web socket 连接成功触发事件
ws.onclose = function () { alert("连接已关闭..."); };

</script>
```

15. HTTP Response 的 Header 里面都有什么？。

内容较多，举几个例子如下

1	Cache-Control: 告诉所有的缓存机制是否可以缓存及哪种类型
2	Content-Length: 响应体的长度
3	Content-Type: 返回内容的 MIME 类型，所谓的 MIME 就是解析的是什么，比方说: jpg 就是图片
4	Expires: 响应过期的日期和时间
5	Set-Cookie: 设置 Http Cookie

6.浏览器相关

1. 为什么会有跨域的问题出现？

跨域,指的是浏览器不能执行其他网站的脚本，它是由浏览器的同源策略造成的,是浏览器对 javascript 施加的安全限制，防止他人恶意攻击网站

比如一个黑客,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和密码登录时,如果没有同

源限制,他的页面就可以通过 JavaScript 读取到你的表单中输入的内容,这样用户名和密码就轻松到手了。

2. 前端安全 XSS,CSRF 这些是什么？

XSS: 跨站脚本攻击。xss 攻击的主要目的是想办法获取目标攻击网站的 cookie,因为有了 cookie 相当于有了 session。

恶意攻击者往 Web 页面里插入恶意 Script 代码,当用户浏览该网页之时,嵌入其中 Web 里面的 Script 代码会被执行,从而达到恶意攻击用户的目的,避免采取的措施:编码、过滤、校验

csrf: 跨站点伪装请求,CSRF 攻击者在用户已经登录目标网站之后,诱使用户访问一个攻击页面,利用目标网站对用户的信任,以用户身份在攻击页面对目标网站发起伪造用户操作的请求,达到攻击目的。防御手段:1. 尽量使用 POST,限制 GET,2. 加验证码

3. 浏览器如何加载页面的,script 脚本阻塞有什么解决办法,defer 和 async 的区别是什么？

从浏览器地址栏的请求链接开始,浏览器通过 DNS 解析查到域名映射的 IP 地址,成功之后浏览器端向此 IP 地址取得连接,成功连接之后,浏览器端将请求信息通过 HTTP 协议向此 IP 地址所在服务器发起请求,服务器接收到请求之后等待处理,最后向浏览器端发回响应,此时在 HTTP 协议下,浏览器从服务器接收到 text/html 类型的代码,浏览器开始显示此 html,并获取其中内嵌资源地址,然后浏览器再发起请求来获取这些资源,并在浏览器的 html 中显示

1 推迟加载（延迟加载）

如果页面初始的渲染并不依赖于 js 或者 CSS 可以用推迟加载,就是最后在加载 js 和 css,把引用外部文件的代码写在最后

2 defer 延迟加载

`<script src="" defer></script>`在文档解析完成开始执行,并且在 DOMContentLoaded 事件之前执行完成,会按照他们在文档出现的顺序去下载解析。效果和把 script 放在文档最后`</body>`之前是一样的。

注: defer 最好用在引用外部文件中使用,用了 defer 不要使用 `document.write()` 方法;使用 defer 时最好不要请求样式信息,因为样式表可能尚未加载,浏览器会禁止该脚本等待样式表加载完成,相当于样式表阻

塞脚本执行

3

异步加载

async 异步加载：就是告诉浏览器不必等到加载完外部文件，可以边渲染边下载，什么时候下载完成什么时候执行。`<script type="text/javascript" src="a.js" async></script>`

defer 和 async 的区别：`<script async src="example.js"></script>`有了 async 属性，表示后续文档的加载和渲染与 js 脚本的加载和执行是并行进行的，即异步执行；`<script defer src="example.js"></script>`

有了 defer 属性，加载后续文档的过程和 js 脚本的加载(此时仅加载不执行)是并行进行的(异步)，js 脚本的执行需要等到文档所有元素解析完成之后，DOMContentLoaded 事件触发执行之前

4. 浏览器强制缓存和协商缓存是什么？

强制缓存：是利用 http 的返回头中的 Expires 或者 Cache-Control 两个字段来控制的，用来表示资源的缓存时间

协商缓存：就是由服务器来确定缓存资源是否可用，所以客户端与服务器端要通过某种标识来进行通信，从而让服务器判断请求资源是否可以缓存访问，这主要涉及到下面两组 header 字段，这两组搭档都是成对出现的，即第一次请求的响应头带上某个字段（Last-Modified 或者 Etag），则后续请求则会带上对应的请求字段（If-Modified-Since 或者 If-None-Match），若响应头没有 Last-Modified 或者 Etag 字段，则请求头也不会有对应的字段

5. 浏览器的全局变量有哪些？

有：alert, location, open(), setTimeout(), clearInterval()等

6. 浏览器同一时间能够从一个域名下载多少个资源？

一般是限制在 10 个以内

7. 按需加载，不同页面的元素判断标准是怎么样？

访问的数据量过大的时候用缓存明显不太合适的时候。可以用按需加载。
如果是数据量不是很多的话可以放在

8. WEB 存储, COOKIES, LOCALSTORAGE 等的使用规则和区别?

web 存储就指的是本地存储, 包括 localStorage 和 sessionStorage

Cookies: cookie 在浏览器与服务器之间来回传递, cookie 只在设置的 cookie 过期时间之前一直有效, 即使窗口或浏览器关闭, cookie 数据不能超过 4k。

localStorage: localStorage 不把数据发给服务器, 仅在本地保存, 始终有效, 长期保存, 可以达到 5M 或更大存储大小

9. 浏览器内核你都知道哪些?

火狐浏览器: Mozilla Firefox, 内核是 Gecko

opera 浏览器: 内核是 blink

Safari 浏览器: 使用的是苹果公司自己的内核: webkit

一些国内的浏览器他们的内核

搜狗浏览器:	兼容模式 (IE: Trident) 和高速模式 (webkit)
傲游浏览器:	兼容模式 (IE: Trident) 和高速模式 (webkit)
QQ 浏览器:	普通模式 (IE: Trident) 和极速模式 (webkit)
360 极速浏览器:	基于谷歌 (Chromium) 和 IE 内核
360 安全浏览器:	IE 内核

10. 什么是预加载, 懒加载?

预加载: 提前加载图片, 当用户需要查看时可直接从本地缓存中渲染

懒加载: 也就是延迟加载。当访问一个页面的时候, 先把 img 元素或是其他元素的背景图片路径替换成一张大小为 1*1px 图片的路径 (这样就只需请求一次, 俗称占位图), 只有当图片出现在浏览器的可视区域内时, 才设置图片真正的路径, 让图片显示出来。这就是图片懒加载

11. 一个 XMLHttpRequest 实例它有几种状态分别代表什么？

XMLHttpRequest 的几种状态	
0	对象没有完成初始化
1	对象开始发送请求
2	对象的请求发送完成
3	对象开始读取服务器响应
4	对象读取服务器响应结束

12. DNS 解析原理，输入网址之后如何查找服务器？

地 DNS 服务器一般都是你的网络接入服务器商提供，比如中国电信，中国移动。

查询 www.163.com 的 DNS 请求到达本地 DNS 服务器之后，本地 DNS 服务器会首先查询它的缓存记录，如果缓存中有此条记录，就可以直接返回结果。如果没有，本地 DNS 服务器还要向 DNS 根服务器进行查询。

根 DNS 服务器没有记录具体的域名和 IP 地址的对应关系，而是告诉本地 DNS 服务器，你可以到域服务器上去继续查询，并给出域服务器的地址。

本地 DNS 服务器继续向域服务器发出请求，在这个例子中，请求的对象是.com 域服务器。.com 域服务器收到请求之后，也不会直接返回域名和 IP 地址的对应关系，而是告诉本地 DNS 服务器，你的域名的解析服务器的地址。

最后，本地 DNS 服务器向域名的解析服务器发出请求，这时就能收到一个域名和 IP 地址对应关系，本地 DNS 服务器不仅要把 IP 地址返回给用户电脑，还要把这个对应关系保存在缓存中，以备下次别的用户查询时，可以直接返回结果，加快网络访问

13. 服务器如何识别是你在操作，说说思路？

- ①当浏览器首次访问服务器时,服务器会为客户端创建一个 session（每个用户独有的房间，用来存放这个对象的相关信息和内容），并通过特殊算法算出一个 sessionId（类似于双方都知道的唯一暗号），用来标识该 session 对象。
- ②当浏览器再次（session 还在有效期内）向服务器请求资源的时候，浏览器将 sessionId 和请求内容一起发送到

服务端。服务端通过对比自身存储的 sessionId 来判断用户之前是否存在，并返回对应的内容给不同用户。

③因为标识符存在内存里，所以当浏览器关闭时，浏览器保存的 sessionId 就会消失。服务器将匹配失败，默认为此请求是新用户提出的，如上文顺序，重新创建一个 session 容器，和相应的唯一 sessionId，返回给浏览器。

一分钟专业解释：

① 服务器在响应头内加上” Set-Cookie:XXXXXXXXXXXX “(相当于一个唯一的 ID 符),此信息是服务器随机生成的，放在服务器内存里，不会重复,这就是 sessionId。

②当浏览器得到这个 sessionId 会把它放在自己的进程内存里,.然后你继续发请求给这个网站的时候,浏览器就会把这个 sessionId 放在请求头里发送给该服务器了,这样服务器得到 sessionId 后再和自己内存里存放的 sessionId 对比锁定客户端,从而区分不同客户端,完成会话.

③关闭浏览器结束进程,则这个 sessionId 将消失,如果用户又打开浏览器想继续这次会话的时候,就会因为发送的请求中没有这个 sessionId，而使服务器无法辨别请求身份。

14. 浏览器的渲染流程你了解吗？

1	解析 HTML 文件，创建 DOM 树
2	解析 CSS：优先级：浏览器默认设置<用户设置<外部样式<内联样式<HTML 中的 style 样式
3	将 CSS 与 DOM 合并，构建渲染树
4	布局和绘制，重绘（repaint）和重排（reflow）
重排：若渲染树的一部分更新，且尺寸变化，就会发生重排；	
重绘：部分节点需要更新，但不改变其他集合形状。如改变某个元素的颜色，就会发生重绘	

15. 介绍几个 IE 浏览器的兼容问题？

- 1 块属性标签 float 之后，又有横向的 margin 值，在 IE6 中显示会比设置的大（IE6 双边距 bug）
- 2 设置较小的高度标签（一般小于 10px），在 IE6，IE7，会超出自己设置的高度，**解决办法：设置较小的高度标签（一般小于 10px），在 IE6，IE7，遨游中超出自己设置的高度**

3 图片默认有间距，**解决方案：使用 float 为 img 布局**

4 给一个元素设置了高度和宽度的同时，还为其设置 margin 和 padding 的值，会改变该元素的实际大小。**解决办法：在需要加 margin 和 padding 的 div 内部加一个 div,在这个 div 里设置 margin 和 padding 值**

16. 对于 Session 你知道哪些内容？

session 是存放在服务器的内存中里，所以 session 里的数据不断增加会造成服务器的负担，所以会把很重要的信息存储在 session 中，session 的信息是通过 sessionid 获取的，而 sessionid 是存放在会话 cookie 当中的，当浏览器关闭的时候会话 cookie 消失，所以 sessionid 也就消失了，但是 session 的信息还存在服务器端。一般 session 是和 cookie 结合起来使用的

17. 如何实现一个拖拽，说一下思路？（表述方式需要修改）

基本思路如下，代码省略：拖拽状态 = 0 鼠标在元素上按下时候{

拖拽状态 = 1

记录下鼠标的 x 和 y 坐标

记录下元素的 x 和 y 坐标

}

鼠标在元素上移动的时候{

如果拖拽状态是 0 就什么也不做。

如果拖拽状态是 1，那么

元素 y = 现在鼠标 y - 原来鼠标 y + 原来元素 y

元素 x = 现在鼠标 x - 原来鼠标 x + 原来元素 x

}

鼠标在任何时候放开的时候{

```
拖拽状态 = 0
```

```
}
```

18. 拆解一下 URL 的各个部分，分别是什么意思？

例如：scheme://host:port/path?query#fragment

1	.scheme:通信协议，常用的 http,ftp,maito 等
2	.host:主机，服务器(计算机)域名系统 (DNS) 主机名或 IP 地址
3	.port:端口号，整数，可选，省略时使用方案的默认端口，如 http 的默认端口为 80
4	.path:路径，由零或多个"/"符号隔开的字符串，一般用来表示主机上的一个目录或文件地址
5	.query:查询，可选，用于给动态网页传递参数，可有多参数，用"&"符号隔开，每个参数的名和值用"="符号隔开
6	.fragment:信息片断，字符串，用于指定网络资源中的片断。例如一个网页中有多个名词解释，可使用 fragment 直接定位到某一名词解释。(也称为锚点)

7.前端工程化常用手段

1. Webpack,gulp,grunt 等构建工具了解多少，它们有什么区别？

Webpack 与 Gulp、Grunt 没有什么可比性，Webpack 可以看作模块打包机，通过分析你的项目结构，找到 JavaScript 模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript 等），并将其转换和打包为合适的格式供浏览器使用。主要用于模块化方案，预编译模块的方案。

gulp 是工具链、构建工具，可以配合各种插件做 js 压缩，css 压缩，less 编译 替代手工实现自动化工作，主要是：

1.构建工具 2.自动化 3.提高效率用。相比于 grunt 的频繁 IO 操作，gulp 的流操作，能更快地更便捷地完成构建工作。

Gulp/Grunt 是一种能够优化前端的开发流程的工具，而 WebPack 是一种模块化的解决方案，不过 Webpack 的优点

使得 Webpack 在很多场景下可以替代 Gulp/Grunt 类的工具。

Gulp 较之 grunt 的优势

1	易用， Gulp 相比 Grunt 更简洁，而且遵循代码优于配置策略，维护 Gulp 更像是写代码。
2	高效， Gulp 相比 Grunt 更有设计感，核心设计基于 Unix 流的概念，通过管道连接，不需要写中间文件。
3	高质量， Gulp 的每个插件只完成一个功能，各个功能通过流进行整合并完成复杂的任务。例如：Grunt 的 imagemin 插件不仅压缩图片，同时还包括缓存功能。而在 Gulp 中，缓存是另一个插件，可以被别的插件使用，这样就促进了插件的可重用性。
4	易学， Gulp 的核心 API 只有 5 个，掌握了 5 个 API 就学会了 Gulp，之后便可以通过管道流组合自己想要的任务。
5	代码优于配置，维护 Gulp 更像是写代码，而且 Gulp 遵循 CommonJS 规范，因此跟写 Node 程序没有差别。

2. Webpack,的入口文件如何配置？

```
const path = require('path');
module.exports={
  //入口文件的配置项
  entry:{
    entry: './src/entry.js'
  },
  //出口文件的配置项
  output:{
    //输出的路径，用了Node语法
    path:path.resolve(__dirname, 'dist'),
    //输出的文件名称
    filename: 'bundle.js'
  },
  //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:{}
}
```

3. Webpack 的 loader 和 plugins 的区别？

loader 用于加载待打包的资源，plugin 用于扩展 webpack。

loader 用于加载某些资源文件。因为 webpack 本身只能打包 commonjs 规范的 js 文件，对于其他资源例如 css，图片，或者其他的语法集，比如 jsx，coffee，是没有办法加载的。这就需要对应的 loader 将资源转化，加载进来。从字面意思也能看出，loader 是用于加载的，它作用于一个个文件上。

plugin 用于扩展 webpack 的功能。它直接作用于 webpack，扩展了它的功能。当然 loader 也变相的扩展了 webpack，但是它只专注于转化文件（transform）这一个领域。而 plugin 的功能更加的丰富，而不仅局限于资源的加载。



提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

8 模块化相关

1. 对 AMD,CMD,CommonJs 有什么理解？

AMD 规范即异步模块加载机制。从规范描述页面看，AMD 很短也很简单，但却完整描述了模块的定义，依赖关系，引用关系以及加载机制。AMD 规范其实只有一个主要接口 `define(id,dependencies,factory)`，它要在声明模块的时候指定所有的依赖 `dependencies`，并且还要当做形参传到 `factory` 中，对于依赖的模块提前执行，依赖前置

```
1.   define("module", ["dep1", "dep2"], function(d1, d2) {  
2.     return someExportedValue;  
3. });  
4. require(["module", "../file"], function(module, file) { /* ... */ });
```

优点：1、适合在浏览器环境异步加载 2、并行加载多个模块

缺点：1、提高开发成本，代码阅读和书写比较困难 2、不符合通用的模块思维方式，是一种妥协的实现

实现：requireJS, NodeJs, Dojo, JQuery

CMD 规范和 AMD 相似，尽量保持简单，并且与 CommonJS 和 NodeJS 的 Modules 规范保持了很大的兼容性。在 CMD 中，一个模块就是一个文件，格式为：define(factory)

```
1. define("module", ["dep1", "dep2"], function(d1, d2) {  
2.   return someExportedValue;  
3. });  
4. require(["module", "../file"], function(module, file) { /* ... */ });
```

优点：1、依赖就近，延迟执行 2、很容易在 node 中运行

缺点：1、依赖 SPM 打包，模块的加载逻辑偏重

实现：SeaJS

CommonJS 是在浏览器环境之外构建 JavaScript 生态系统为目标产生的项目，比如服务器和桌面环境中。CommonJS 规范是为了解决 JavaScript 的作用域问题而定义的模块形式，可以使每个模块在它自身的命名空间中执行。该规范的主要内容是：模块必须通过 module.exports 导出对外的变量或接口，通过 require() 来导入其他模块的输出到当前模块。

```
1. // moduleA.js  
2. module.exports = function( value ){  
3.   return value * 2;  
4. }
```

```
1. // moduleB.js  
2. var multiplyBy2 = require('./moduleA');
```

3. `var result = multiplyBy2(4);`

CommonJS 是同步加载模块，一个单独的文件就是一个模块。但其实也有浏览器端的实现，其原理是将所有模块都定义好并通过 id 进行索引，这样就可以浏览器进行解析了 **服务器端的 Node.js 遵循 CommonJS 规范**。核心思想是允许模块通过 `require` 方法来同步加载所要依赖的其他模块，然后通过 `exports` 或 `module.exports` 来导出需要暴露的接口。

1. `require("module");`

2. `require("../file.js");`

3. `exports.doStuff = function() {};`

4. `module.exports = someValue;`

优点：1、服务器端便于重用 2、NPM 中已经将近 20w 个模块包 3、简单并容易使用


缺点：1、同步的模块方式不适合在浏览器环境中，同步意味着阻塞加载，浏览器资源是异步加载的 2、不能非阻塞的并行加载多个模块

区别：（AMD or CMD）

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
3	AMD 的 API 默认是一个当多个用，CMD 的 API 严格区分，推崇职责单一。
4	CMD 相当于按需加载，定义一个模块的时候不需要立即制定依赖模块，在需要的时候 <code>require</code> 就可以了，比较方便；而 AMD 则相反，定义模块的时候需要制定依赖模块。


 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

2. 为什么要模块化？不用的时候和用 RequireJS 的时候代码该如何书写？

1	方便大量的 js 脚本代码的管理维护以及团队配合开发
2	有效解决命名空间冲突及文件依赖加载顺序问题
3	有利于模块的版本管理，提高可维护性，有利于前端性能优化，跨环境共享模块
 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！	

3. 分别说说同步和异步模块化的应用场景，说一下 AMD 异步模块化实现的原理？

引入 JS 时会遇到需要异步加载文件，此时 `require.async` 便可满足异步加载需求

 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

9. 框架 / 类库

1. 说一说你都使用了什么框架？

UI 框架	bootstrap、Mui、layui
js 框架	vue、angular、react

 面试的时候你只需要说出你常用得就行，不要都数，再结合你的简历得项目说其中的一个，详谈

2. zepto 和 jquery 是什么关系，他们之间有什么联系？

二者都是 js 库，zepto 是 jQuery 的轻量级替代品，它与 jquery 有着类似的 api，zepto 主要是用在移动端，不支持 IE 浏览器，jquery 主要用在 PC 端，jQuery 版本 2 以上不支持 IE6，7，8 浏览器

3. jquery 源码如何实现选择器，为什么\$取得对象要设计成数组的形式，这样设计有什么目的了？

jquery 内部采用了一种“类数组对象”的方式作为存储结构，既可以像对象一样处理 jQuery 操作，也可以

像数组一样使用 `push`、`pop`、`shift`、`sort`、`each`、`map` 等类数组的方法操作 jquery 对

目的：

4. jquery 如何绑定事件，有几种类型和区别？

jquery 绑定事件的方法分别是：bind(),live(),delegate()和 on()，

像 bind(),live(),delegate(),随着 jquery 版本的更新，已经被移除，注意：bind()是在 3.0 版本之后被移除

的，现在用的最多的是 on()，on()既可以绑定单事件，也可以绑定多事件，还可以进行事件委托

区别就是：bind()的事件绑定是只对当前页面选中的元素有效，对动态创建的元素 bind()事件，是没有办法达到效果的，而其余三个可以

5. 什么是 MVVM,MVC,MVP？

介绍这种开发模式切记：数据驱动视图是核心，这三种模式都是为了方便后期的代码的维护，

MVC：先介绍 M:Model(模型)，V:View(视图)，C: Controller (控制器),View 层是展示 html 页面的，

Controller 层是业务逻辑，Model 层是数据保存，视图的改变会通过控制器要求数据层改变状态，然后反馈给视图层。

MVVM：是将 Controller 改为 VM，是 Model-View-ViewModel 的缩写，视图 (View) 可以独立于 Model 变化和修改，一个 ViewModel 可以绑定到不同的"View"上，当 View 变化的时候 Model 可以不变，当 Model 变化的时候 View 也可以不变。

MVP：MVP 模式是将 Controller 改为 Presenter，View 和 Model 之间不发生联系，都通过 Presenter 传递，所有的交互都发生在 Presenter 内部，而在 MVC 中 View 会直接从 Model 中读取数据而不是通过 Controller

6. Vue 和 Angular 的双向数据绑定原理？

Vue 双向数据绑定的原理：

vue 将普通的对象的属性通过 `Object.defineProperty` 转换为 ES5 特性之一的 `getter/setter`，模板中每个指令/数据绑定都有一个对应的 `watcher` 对象，当修改对象值的时，首先会触发属性的 `setter`，在 `setter` 被调用时，会触发 `watcher` 重新计算，也就会导致它的关联指令更新 DOM

```
// Vue 双向数据绑定的原理
function defineReactive(obj, key, value) {
  var dep = new Dep()
  Object.defineProperty(obj, key, {
    enumerable: true,
    configurable: true,
    get: function reactiveGetter() {
      if (Dep.target) {
        dep.depend()
      }
      return value
    },
    set: function reactiveSetter(newVal) {
      if (value === newVal) {
        return
      } else {
        value = newVal
        dep.notify()
      }
    }
  })
}
```

Angular 双向数据绑定的原理：

angular.js 是通过脏值监测的方式查看数据是否变更，最简单的方法是通过 `setInterval()` 定时循环检测数据变动

7. Vue 和 Angular 的组件通信以及路由原理？

vue 得组件通信，

1) 父组件传递数据给子组件

举个栗子

```
// 父组件:
<parent>
  <child :child-msg="msg"></child> //这里必须要用 - 代替驼峰
</parent>

data(){
  return {
    msg: [1,2,3]
  };
}
```

子组件通过 props 来接收数据:

方式 1:

props: ['childMsg']

子组件与父组件通信

```
//子组件:
<template>
  <div @click="up"></div>
</template>

methods: {
  up() {
    this.$emit('upup', 'hehe'); //主动触发upup方法, 'hehe'为向父组件传递的数据
  }
}

//父组件:
<div>
  <child @upup="change" :msg="msg"></child>
  //监听子组件触发的upup事件,然后调用change方法
</div>
methods: {
  change(msg) {
    this.msg = msg;
  }
}
```

*非父子组件通信

这时可以通过eventBus来实现通信。

所谓eventBus就是创建一个事件中心，相当于中转站，可以用它来传递事件和接收事件。

let bus = new Vue(); //创建事件中心

//组件1触发:

```
<div @click="eve"></div>
```

```
methods: {
```

```
  eve() {
```

```
    bus.$emit('change','hehe'); //bus触发事件
```

```
  }
```

```
}
```

组件2接收:

```
<div></div>
```

```
created() {
```

```
  bus.$on('change', () => { //bus接收事件
```

```
    this.msg = 'hehe';
```

```
  });
```

```
}
```

Angular 组件间通信

Angular 组件之间的通信方式分为三种:

1) 父组件向子组件通信（输入属性）分为两步:

1、在子组件上定义自己要接受父组件输入的参数:

```
@Input()
```

```
stockCode: string;
```

```
@Input()
```

```
amount: number;
```

```
//@Input()修饰器表示stockCode、amount是要接受父组件传入属性的变量
```

2、在父组件的 HTML 模板中声明要输入的数据:

```
<app-bind [stockCode]="stock" [amount]="data">
```

```
  //主要输入属性的name要一致，<app-bind>是子组件
```

```
</app-bind>
```

子

2) 子组件向父组件通信（输出属性）

1、在子组件中设置要输出的类型:

```
@Output()
```

```
lastPrice: EventEmitter<PriceQuote> = new EventEmitter();
```

```
//@Output()修饰器修饰要输出的变量
```

```
//PriceQuote这个泛型指的是：输出的变量类型
```

2、在 html 模板中设置事件：

```
<app-price-quote (lastPrice)="priceQuoteHandler($event)">  
  参数解析  
    lastPrice 是在子组件中设置的输出变量的name  
    <app-price-quote>是子组件  
    $event 保留了事件触发的dom的所有属性  
</app-price-quote>
```

3、在父组件的控制器中接收输出属性

```
声明一个和输出变量类型相同的变量/  
priceQuote: PriceQuote = new PriceQuote('', 0);  
  
priceQuoteHandler(event: PriceQuote) {  
  this.priceQuote = event;  
} //事件，用于接收输出属性
```

3) 使用中间人模式通信

所谓中间人模式就是：当 A =》 C 通信时，先将 A =》 B，B=》 C，

B 就是所谓中间人模式；

按照情况一般分为两种：

1、兄弟组件通信

这里我们选择父组件做为中间人，先是使用输出属性，再使用输入属性就 OK 了

2、非兄弟组件通信

选择一个依赖注入 service 做为中间人，然后进行通信。

10. NodeJs

1. 对 node.js 有没有了解

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

node.js...它既是开发平台，也是运行环境，也是个新的语言...它本身是基于 google 的 javascript v8 引擎开发的，因此在编写基于它的代码的时候使用 javascript 语言。但是又不同于传统概念的 javascript...它的服务端功能以及部分客户端功能必须在服务端运行，所以它实际上是一种在服务端的开发+运行的 javascript 语言。有一点类似于 Perl + PHP 或者 Python 的概念。它本身可以作为 HTTP Server，也可以当作 TCP Server 用。

2 .Express 和 koa 有什么关系，有什么区别

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

koa 是由 Express 原班人马打造的，致力于成为一个更小、更富有表现力、更健壮的 Web 框架。使用 koa 编写 web 应用，通过组合不同的 generator，可以免除重复繁琐的回调函数嵌套，并极大地提升错误处理的效率。

koa 不在内核方法中绑定任何中间件，它仅提供了一个轻量优雅的函数库，使得编写 Web 应用变得得心应手。

koa 是一个比 express 更精简，使用 node 新特性的中间件框架，相比之前 express 就是一个庞大的框架如果你喜欢 DIY 很潮，可以考虑 koa，他有足够的的扩展和中间件，而且自己写很简单。

如果你想简单点，找一个框架啥都有，那么先 express

3 .node.js 适合做什么业务

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

Nodejs 是单线程，非阻塞 I/O，事件驱动，它的特点决定了它适合做一些大量 I/O 的东西，比如，聊天室，表单提交等不需要大量计算的功能。做一些微信后端开发，或者做消息系统等。可以整个项目用，也可以根据它的特点在某个模块使用，比如 socketio，打造一个消息系统等

4 .node.js 与 php 和 Java 的区别

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

Node.js 是前端中的一种技术，是单线程，非阻塞 I/O，事件驱动，是一门很强大的技术。

java，一直很牛 X，企业级别的项目，基本上都用这个，如果考虑长期发展，这个是首选，不过同时这个难度也是最大的，如果自己语言基础，这个和后面的一个，最好不要选择，你选择测试可能会更加好一点。

PHP 是一门脚本语言，基本都用在 web 应用中的中间层，负责数据库以及前台页面交互和信息传递。

5. Nodejs 中的 Stream 和 Buffer 有什么区别

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

Buffer：为数据缓冲对象，是一个类似数组结构的对象，可以通过指定开始写入的位置及写入的数据长度，往其中写入二进制数据。

Stream：是对 buffer 对象的高级封装，其操作的底层还是 buffer 对象，stream 可以设置为可读、可写，或者即可读也可写，在 nodejs 中继承了 EventEmitter 接口，可以监听读入、写入的过程。具体实现有文件流，httpresponse 等。

6. node 的异步问题是如何解决的

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

模块化：将回调函数转换为独立的函数

使用流程控制库，例如 async

使用 Promise

使用 `async/await`(参考 `Async/Await` 替代 `Promise` 的 6 个理由)

关于如何解决具体的异步问题你可以(扫一扫)



(来自: SegmentFault)



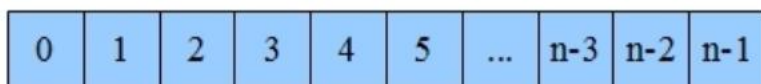
(来自: CSDN)

11. 数据结构相关

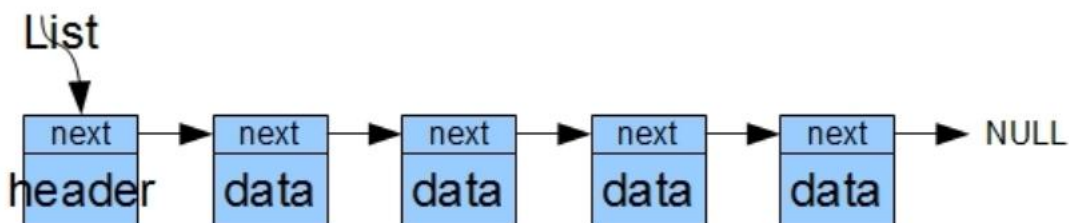
1. 基本数据构: (数组、 对列、 链表、堆、二叉树、嘻哈表等等)

你可以这样回答 (提示: 回答该问题建议: 请理解内容, 使用自己的语言表达为最佳的回答哦!)

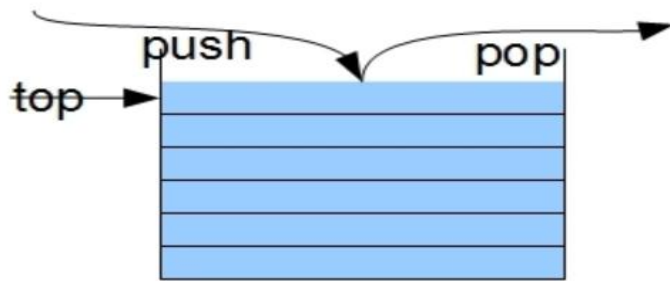
数组: 数组是最最基本的数据结构, 很多语言都内置支持数组。数组是使用一块连续的内存空间保存数据, 保存的数据的个数在分配内存的时候就是确定的。(如图所示)



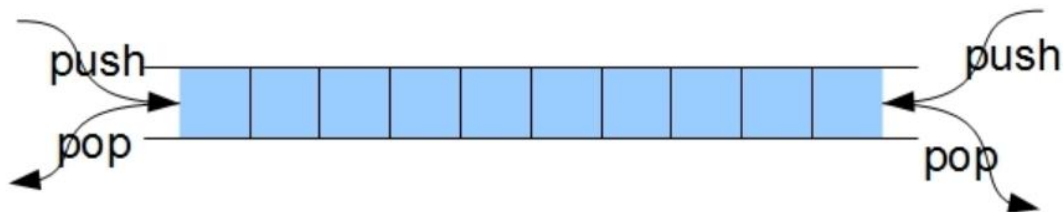
链表: 存储的数据在地址空间上可连续, 可不连续, 链表中的每一个节点都包括数据和指向下一个地址的指针, 查找数据的时间复杂度为 $O(n)$, 方便数据的增删。



栈: 栈是一种先入后出的逻辑结构, 每次加入新的元素和拿走元素都在顶部操作。



对列：是一种先入后出的逻辑结构，对于元素的操作分别在队头和队尾，元素的插入在队尾，元素的删除在队头。

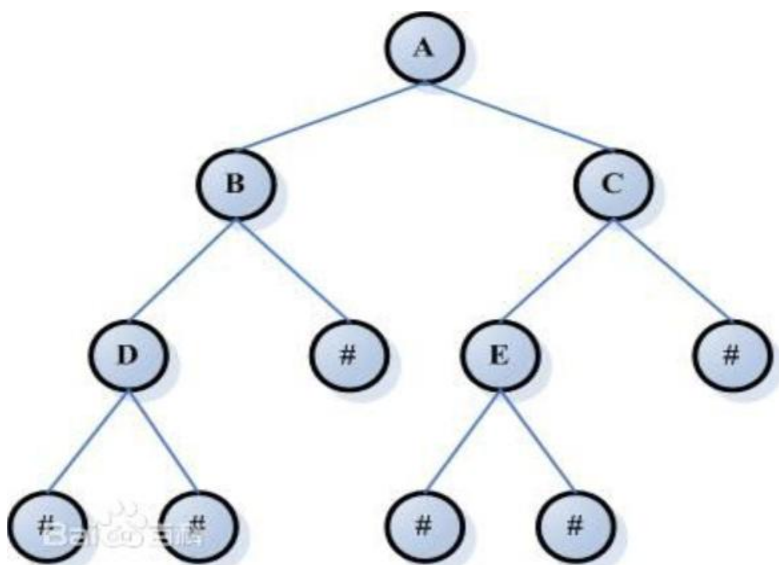


二叉树：每个节点至多只有两个子树的结构，在父节点中有指向左右子树的指针。

二叉树的 先序遍历：根 - 左 - 右。中序遍历：左 - 根 - 右。后序遍历：左 - 右 - 根。

查找二叉树：左子树的值小于根节点的值，右子树的值大于根节点的值，在插入数据时，从根节点开始往下比较，小于比较值则放在左边，大于比较值放在右边。插入一个值的时间复杂度是 $O(\log n)$ 。

平衡二叉树：左右子树的高度差的绝对值不超过 1。



2. 8 种排序算法，原理，以及适用的场景和复杂度。

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

各种常用排序算法						
类别	排序方法	时间复杂度			空间复杂度	稳定性
		平均情况	最好情况	最坏情况	辅助存储	
插入排序	直接插入	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	shell排序	$O(n^{1.3})$	$O(n)$	$O(n^2)$	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
	堆排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
	快速排序	$O(n\log_2n)$	$O(n\log_2n)$	$O(n^2)$	$O(n\log_2n)$	不稳定
归并排序		$O(n\log_2n)$	$O(n\log_2n)$	$O(n\log_2n)$	$O(1)$	稳定
基数排序		$O(d(r+n))$	$O(d(n+rd))$	$O(d(r+n))$	$O(rd+n)$	稳定

注：基数排序的复杂度中，r代表关键字的基数，d代表长度，n代表关键字的个数

这里就体现 2 个常问的问题：冒泡排序的原理：

S1：从待排序序列的起始位置开始，从前往后依次比较各个位置和其下一位置的大小并执行 S2。

S2：如果当前位置的值大于其下一位置的值，就把他俩的值交换（完成一次全序列比较后，序列最后位置的值即此序列最大值，所以其不需要再参与冒泡）。

S3：将序列的最后位置从待排序序列中移除。若移除后的待排序序列不为空则继续执行 S1，否则冒泡结束。

在举个栗子

```

var len = array.length;
var flag = true;
while (flag) {
    flag = false;
    for (int i = 0; i < len - 1; i++) {
        if (array[i] > array[i + 1]) {
            int temp = array[i + 1];
            array[i + 1] = array[i];
            array[i] = temp;
            flag = true;
        }
    }
    len--;
}

```

快速排序：快速排序是对冒泡排序的一种改进。基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此实现整个数据变成有序序列

在举个栗子：

```

function quickSort(arr){
    //如果数组<=1,则直接返回
    if(arr.length<=1){return arr;}
    var pivotIndex=Math.floor(arr.length/2);
    //找基准，并把基准从原数组删除
    var pivot=arr.splice(pivotIndex,1)[0];
    //定义左右数组
    var left=[];
    var right=[];
    //比基准小的放在left，比基准大的放在right
    for(var i=0;i<arr.length;i++){
        if(arr[i]<=pivot){
            left.push(arr[i]);
        }
        else{
            right.push(arr[i]);
        }
    }
    //递归
    return quickSort(left).concat([pivot],quickSort(right));
}

```

更多排序扫这里啦~~~



(扫我查看更多)

3. 说出越多越好的费波拉切数列的实现方法？

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

1) 递归的方法：

```
function fib(n){  
    if(n==1||n==2){  
        return 1;  
    }  
    return fib(n-1)+fib(n-2);  
}  
fib(10);
```

非递归的方法：

```
function fb(n){  
    var a,b,res;  
    a = b = 1;  
    for(var i=3;i<=n;i++){  
        res = a + b;  
        a = b;  
        b = res;  
    }  
    return res;  
}  
fb(10);
```

12. 性能优化相关

1. CDN 的用法是什么？什么时候用到

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

CDN 即内容分发网络。CDN 的基本原理是广泛采用各种缓存服务器，将这些缓存服务器分布到用户访问相对集中的地区或网络中，在用户访问网站时，利用全局负载技术将用户的访问指向距离最近的工作正常的缓存服务器上，由缓存服务器直接响应用户请求

CDN 的使用，以腾讯 CDN 为例：

- 1 登录腾讯云 CDN，选择“接入管理”，点击“添加域名”添加我们需要加速的域。
- 2 进入“添加域名”界面，在域名那里填好你要加速的域名。源站 IP 那里可以填自己服务器的 IP。如果用的是虚拟主机，可以填空间商给你的域名。配置完成后点下一步！
- 3 进入“基本配置”界面，填写相关要求，在缓存时间那里，腾讯 CDN 默认的是将你所有的文件缓存 3 天，你可以根据自身情况修改天数。填写好后点击下一步。
- 4 下一步也差不多，填写相关要求，没问题点击提交就好。

CDN 的适用场景：

解决因分布、带宽、服务器性能带来的访问延迟问题,适用于网站站点/应用加速、点播、直播、视音频点播、大文件下载分发加速、移动应用加速等场景

2. CDN 的用法是什么？什么时候用到

你可以这样回答（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦!）

- 1 把 css 放在 head 中加载，这个能让页面更早的开始渲染。避免把 css 放在页面尾部，否则可能会出现闪屏，如果 css 加载的很慢，DOM 结构先出来，css 后出来，然后页面样式突然发生变化，就造成了闪屏。
- 2 把 js 放在 body 末尾加载，因为 js 它本身会阻塞 HTML 的解析以及 css 的渲染。

- 3 不要使用 css 的表达式，css 的表达式一方面是兼容性问题，虽然看起来比较强大，但是实际性能开销很大，因为它实际的执行频率是远远超出预期的，如果使用了 css 的表达式，可能导致页面卡顿。
- 4 用外链的方式引用 css 和 js，可以有效的减少 HTML 的体积，并且外链了之后，css 和 js 作为静态资源可以给他设置合适的缓存的响应头；能够合理的利用浏览器的缓存。
- 5 压缩 js 和 css，在生产环境里面删除这一类文件不必要的注释、空白，并且对 js 进行变量名压缩，混淆压缩，对 css 进行属性的合并，然后进行选择符的合并。
- 6 不重复加载 js，因为在 IE 里面，还是会有多个请求，不能发挥缓存优势，并且你重复加载 js，意味着要更长的 js 执行时间。
- 7 让 Ajax 请求可缓存，GZIP、内容压缩都可以适用。
- 8 用 GET 方式发起 Ajax 请求，因为 GET 方式可以缓存，如果是获取信息的，那么 GET 方式是更语义化的。
- 9 组件延迟加载，保障页面关键的静态资源优先加载，因为并发数限制，还有一些延迟加载的典型手段叫做“lazyload”。
- 10 减少 DOM 节点数，DOM 节点这个规则是非常容易理解，就是如果你的 DOM 结构非常的复杂，那么浏览器在解析的时候，进行布局、渲染时计算量更大，那么少一点的话，浏览器的开销会少一些，渲染布局的速度就会快一些。
- 11 避免在页面中使用 frame 类（包括 iframe 和 frameset），因为 iframe 它会阻塞父文档的 onload，即使它是一个空白的也会比较耗时。
- 12 要减少 COOKIE 的体积，因为 COOKIE 在每一次请求，就是跟主文档相关的，只要是同域的，COOKIE 的全部内容都可以带上。
- 13 使用无 cookie 域名加载静态资源，可以减少静态资源加载时的网络传输量，静态资源加载通常是不需要 cookie 的。
- 14 减少 js 中的 DOM 访问，还有对于你查找到的元素，缓存在 js 的变量中，后面就不需要在去查找 DOM 树了，还有节点增加是合理利用 DocumentFragment，然后在把 DocumentFragment 放在实际的 DOM 树里面，然后还有一个是不要用 js 去频繁修改样式。

- 15 使用更多的事件监听机制，基于事件冒泡的委托机制，你可以有效减少绑定的数量。
- 16 使用常见的图片优化手段，对于代码来说，图片的体积很大，常用的图片压缩工具有“PNGCrush”
“JPEGTURAN”“PNGQUANT”。
- 17 不要在 HTML 中缩放图片，你实际上在页面上用到多大图片，那你就提供适当尺寸即可，不然它只会徒增渲染的开销。
- 18 不要把图片 SRC 置空，因为在主文档的浏览器里面都会引发额外的请求。
- 19 任何资源尽量在 25k 以内，因为 iPhone 无法缓存 25k 以上的资源。
- 20 预渲染，预渲染比预下载更进一步，不仅仅下载页面，而且还会预先将它渲染出来，目前在 Chrome
(9.0.597.0) 中有实现，不过需要在 about:flags 中将‘Web Page Prerendering’开启。
- 21 DNS 优化；CDN 优化；http 优化；减少域名解析时间；增多域名提高并发

3.如何优化 dom 操作的性能。

你可以这样回答 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

- 1 查找元素的优化。因为 ID 是唯一的，也有原始的方法，因此使用 ID 查找元素是最快的，其次的是根据类和类型查找元素，通过属性查找元素是最慢的，因此应该尽可能的通过 ID 或者类来查找元素，避免通过属性来查找元素
- 2 减少访问和改变 DOM 元素，包括添加，修改，删除 DOM

改变 DOM 就会引起浏览器渲染，而渲染是相当慢的，因此应该避免不必要的渲染
- 3 减少改变 DOM 的样式类等

改变 DOM 元素的样式，类也会导致浏览器渲染，因此也应该减少不必要的操作
- 4 批量修改 DOM 时从文档流中摘除该元素，对其应用多重改变，将元素带回文档中，这样可以最小化重绘和重排版。

具体方法: 1 隐藏元素, 进行修改, 然后再显示它。2 将原始元素拷贝到一个脱离文档的节点中, 修改副本, 然后覆盖原始元素。

5 减少 iframe iframe 需要消耗大量的时间, 并阻塞下载, 建议少用

6 样式放在 header 中, 脚本放在关闭标签</body>之前样式放在 header 中, 可以加快渲染, 脚本放在关闭标签</body>之前可以加快下载速度, 不免阻塞下载。

7 使用事件委托, 减少绑定事件的数量

8 多次访问同一 DOM, 应该用局部变量缓存该 DOM

4.单页面应用有什么 SEO 方案？。

你可以这样回答 （提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

1 前端要采用 history 模式（HTML5 History API），一般用来写单页应用的框架都有这个模式。

2 要用后端做一套页面，内容和对应的前端页面一模一样，即所谓静态化，我用的 sails 框架即可实现这一点，主流的如 Node.js 的 express 框架也可以，php 应该也行。

3 必须用 nginx 做代理跳转，将搜索引擎识别出后流量引到后端的端口上，看到预先渲染给搜索引擎看的页面，还有 history 模式自带的 404 问题也需要在 nginx 里将 404 转 index.html 或者 rewrite 才可以

4. 单页面应用有什么 SEO 方案？。

你可以这样回答 （提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

1 原因：较之于传统页面，单页应用需要先下载框架（数据 / 模板），然后才能开始加载数据

方案:

- 1 服务器端渲染首屏（SSR 基于 vue 的服务端下载）
- 2 让服务端把首屏的数据渲染在页面上
- 3 进行基本的 css 模板 js 的编译合并
- 4 减少请求次数，使用 gulp 工具，把 css 打包成一个文件, js 打包成一个文件，模板打包成一个 js 文件(\$templateCache) 可以和 js 文件打包成一起（促使模板 JS 文件和 JS 文件一次性请求）
- 5 代码分块，如果首屏不需要的块，就不用加载了
- 6 路由组件懒加载。当打包构建应用时，Javascript 包会变得非常大，影响页面加载。如果我们能把不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应组件，这样更加高效。
- 7 如果有大量图片使用懒加载

13. 其他相关（拔高）

1.正则表达式

你可以这样回答 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

发展及应用：前期，正则表达式被广泛地应用到各种 UNIX 或类似于 UNIX 的工具中，如：Perl 内部集成了正则表达式的功能，如今正则表达式在基于文本的编辑器和搜索工具中依然占据着一个非常重要的地位，在主流的计算机开发语言中（delphi、Scala、PHP、C#、Java、C++、Objective-c、Swift、VB、Javascript、Ruby 以及 Python 等）都有正则表达式的身影。

正则表达式的特点是：

- 1 灵活性、逻辑性和功能性非常强

2 可以迅速地用极简单的方式达到字符串的复杂控制

3 对于刚接触的人来说，比较晦涩难懂

由于正则表达式主要应用对象是文本，因此它在各种文本编辑器场合都有应用，小到著名编辑器 EditPlus，大到 Microsoft Word、Visual Studio 等大型编辑器，都可以使用正则表达式来处理文本内容。

正则表达式应用——实例应用



(正则在线生成工具)



(常用的正则)

2.前端渲染和后端渲染的区别

你可以这样回答 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

前端渲染：

指的是后端返回 JSON 数据，前端利用预先写的 html 模板，循环读取 JSON 数据，拼接字符串（es6 的模板字符串特性大大减少了拼接字符串的成本），并插入页面。

好处：网络传输数据量小。不占用服务端运算资源（解析模板），模板在前端（很有可能仅部分在前端），改结构变交互都前端自己来了，改完自己调就行。

坏处：前端耗时较多，对前端工作人员水平要求相对较高。前端代码较多，因为部分以前在后台处理的交互逻辑交给了前端处理。占用少部分客户端运算资源用于解析模板。

后端渲染：

前端请求，后端用后台模板引擎直接生成 html，前端接受到数据之后，直接插入页面。

好处：前端耗时少，即减少了首屏时间，模板统一在后端。前端（相对）省事，不占用客户端运算资源（解析模板）

坏处：占用服务器资源。

前端渲染与后端渲染对比：

后端渲染：

页面呈现速度：快，受限于用户的带宽

流量消耗：少一点点（可以省去前端框架部分的代码）

可维护性：差（前后端东西放一起，掐架多年，早就在闹分手啦）

seo 友好度：好

编码效率：低（这个跟不同的团队不同，可能不对）

前端渲染：

页面呈现速度：主要受限于带宽和客户端机器的好坏，优化的好，可以逐步动态展开内容，感觉上会更快一点。

流量消耗：多一点点（一个前端框架大概 50KB）当然，有的用后端渲染的项目前端部分也有在用框架。

可维护性：好，前后端分离，各施其职，代码一目明了。

SEO 友好度：差，大量使用 ajax，多数浏览器不能抓取 ajax 数据。

编码效率：高，前后端各自只做自己擅长的东西，后端最后只输出接口，不用管页面呈现，只要前后端人员能力不错，效率不会低。

3.数据库的四大特性，什么是原子性，表的关系

你可以这样回答 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

数据库中事物的四大特性：原子性、一致性、分离性、持久性

原子性 事务的原子性指的是，事务中包含的程序作为数据库的逻辑工作单位，它所做的对数据修改操

作要么全部执行，要么完全不执行。这种特性称为原子性。

事务的原子性要求，如果把一个事务可看作是一个程序，它要么完整的被执行，要么完全不执行。就是说事务的操纵序列或者完全应用到数据库或者完全不影响数据库。这种特性称为原子性。

假如用户在一个事务内完成了对数据库的更新，这时所有的更新对外部世界必须是可见的，或者完全没有更新。前者称事务已提交，后者称事务撤消（或流产）。DBMS 必须确保由成功提交的事务完成的所有操纵在数据库内有完全的反映，而失败的事务对数据库完全没有影响。

一致性事务的一致性指的是在一个事务执行之前和执行之后数据库都必须处于一致性状态。这种特性称为事务的一致性。假如数据库的状态满足所有的完整性约束，就说该数据库是一致的。

一致性处理数据库中对所有语义约束的保护。假如数据库的状态满足所有的完整性约束，就说该数据库是一致的。例如，当数据库处于一致性状态 S1 时，对数据库执行一个事务，在事务执行期间假定数据库的状态是不一致的，当事务执行结束时，数据库处在一致性状态 S2。

分离性 分离性指并发的事务是相互隔离的。即一个事务内部的操作及正在操作的数据必须封锁起来，不被其它企图进行修改的事务看到。

分离性是 DBMS 针对并发事务间的冲突提供的安全保证。DBMS 可以通过加锁在并发执行的事务间提供不同级别的分离。假如并发交叉执行的事务没有任何控制，操纵相同的共享对象的多个并发事务的执行可能引起异常情况。

DBMS 可以在并发执行的事务间提供不同级别的分离。分离的级别和并发事务的吞吐量之间存在反比关系。较多事务的可分离性可能会带来较高的冲突和较多的事务流产。流产的事务要消耗资源，这些资源必须要重新被访问。因此，确保高分离级别的 DBMS 需要更多的开销。

持久性 持久性意味着当系统或介质发生故障时，确保已提交事务的更新不能丢失。即一旦一个事务提交，DBMS 保证它对数据库中数据的改变应该是永久性的，耐得住任何系统故障。持久性通过数据库备份和恢复来保证。

持久性意味着当系统或介质发生故障时，确保已提交事务的更新不能丢失。即对已提交事务的更新能恢复。一旦一个事务被提交，DBMS 必须保证提供适当的冗余，使其耐得住系统的故障。

表的关系：1、一对一关系 2、一对多关系 3、多对多关系

4.你觉得前端体系应该是怎样的

你可以这样回答 提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！

分离： 前台只需要获取数据，用 JS 渲染即可，这样后台不必懂 HTML，前台不必懂后台的模板语法

优点：

- 1：前端 JS 可以做很大部分的数据处理工作，对服务器的压力减小到最小
- 2：后台错误不会直接反映到前台，错误秒接较为友好
- 3：由于后台是很难去探知前台页面的分布情况，而这又是 JS 的强项，而 JS 又是无法独立和服务器进行通讯的。所以单用后台去控制整体页面，又或者只靠 JS 完成效果，都会难度加大，前后台各尽其职可以最大程度的减少开发难度。

5. SEO

你可以这样回答 （提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

什么是 SEO :seo=Search（搜索） Engine（引擎） Optimization（优化），即搜索引擎优化。

从代码优化 到网站的结构优化方面回答。

网站的结构优化方面：

1) . 控制首页链接数量

网站首页是权重最高的地方，如果首页链接太少，没有“桥”，“蜘蛛”不能继续往下爬到内页，直接影响网站收录数量。但是首页链接也不能太多，一旦太多，没有实质性的链接，很容易影响用户体验，也会降低网站首页的权重，收录效果也不好。

因此对于中小型企业网站，建议首页链接在 100 个以内，链接的性质可以包含页面导航、底部导航、锚文字链接等等，注意链接要建立在用户的良好体验和引导用户获取信息的基础之上。

2). 扁平化的目录层次，尽量让“蜘蛛”只要跳转 3 次，就能到达网站内的任何一个内页。扁平化的目录结构，比如：“植物” --> "水果" --> "苹果"、“桔子”、“香蕉”，通过 3 级就能找到香蕉了。

代码优化：

1) .<title>标题：只强调重点即可，尽量把重要的关键词放在前面，关键词不要重复出现，尽量做到每个页面的<title>标题中不要设置相同的内容。

2). <meta keywords>标签：关键词，列举出几个页面的重要关键字即可，切记过分堆砌。

3). <meta description>标签：网页描述，需要高度概括网页内容，切记不能太长，过分堆砌关键词，每个页面也要有所不同。

如果你想了解更多的话建议你扫（6-1）从 0 开始教你如何学习



（我是 6-1）



（扫我查看更多面试题）

6. mysql 和 mongoDB 有什么区别

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

MySQL

- 1) . 关系型数据库
- 2) . 在不同的引擎上有不同 的存储方式。
- 3) . 查询语句是使用传统的 sql 语句，拥有较为成熟的体系，成熟度很高。
- 4) . 开源数据库的份额在不断增加，mysql 的份额页在持续增长。

5) . 缺点就是在海量数据处理的时候效率会显著变慢。

MongoDB

非关系型数据库 (Nosql) ,属于文档型数据库。先解释一下文档的数据库, 即可以存放 xml、json、bson (即 Binary-JSON) 类型系那个的数据。这些数据具备自述性 (self-describing), 呈现分层的树状数据结构。数据结构由键值(key=>value)对组成。

MongoDB 是由 C++语言编写的, 主要是在为 WEB 应用提供可扩展的高性能数据存储解决方案。

存储方式: 虚拟内存+持久化。

查询语句: 是独特的 Mongoddb 的查询方式。

适合场景: 事件的记录, 内容管理或者博客平台等等。

架构特点: 可以通过副本集, 以及分片来实现高可用。

数据处理: 数据是存储在硬盘上的, 只不过需要经常读取的数据会被加载到内存中, 将数据存储在物理内存中, 从而达到高速读写。

成熟度与广泛度: 新兴数据库, 成熟度较低, Nosql 数据库中最为接近关系型数据库, 比较完善的 DB 之一, 适用人群不断在增长。

MongoDB 的优势:

- 1) . 快速! 在适量级的内存的 Mongoddb 的性能是非常迅速的, 它将热数据存储在物理内存中, 使得热数据的读写变得十分快
- 2) . 高扩展。
- 3) . 自身的 Failover 机制。
- 4) . json 的存储格式。
- 5) . 内置 GridFS, 支持大容量的存储。
- 6) . 内置 Sharding, 分片简单。
- 7) . 高扩展。

8) . 海量数据下, 性能优越。

9) . 支持自动故障恢复 (复制集)。

MongoDB 的缺陷:

1) . 不支持事务操作

2) . 占用空间过大。

3) . MongoDB 没有如 MySQL 那样成熟的维护工具。

4) . 无法进行关联表查询, 不适用于关系多的数据。

5) . 复杂聚合操作通过 mapreduce 创建, 速度慢

6) . 模式自由, 自由灵活的文件存储格式带来的数据错误

7) . MongoDB 没有如 MySQL 那样成熟的维护工具, 这对于开发和 IT 运营都是个值得注意的地方

7. click 在 ios 上有 300ms 延迟, 原因如何及如何解决。

你可以这样回答: (提示: 回答该问题建议: 请理解内容, 使用自己的语言表达为最佳的回答哦!)

原因: 2007 年初, 苹果公司在发布首款 iPhone 前夕, 遇到一个问题 —— 当时的网站都是为大屏幕设备所设计的。于是苹果的工程师们做了一些约定, 应对 iPhone 这种小屏幕浏览桌面端站点的问题。这当中最出名的, 当属双击缩放(double tap to zoom)。这也是会有上述 300 毫秒延迟的主要原因。

当用户一次点击屏幕之后, 浏览器并不能立刻判断用户是要进行双击缩放, 还是想要进行单击操作。因此, iOS Safari 就等待 300 毫秒, 以判断用户是否再次点击了屏幕。

于是, 300 毫秒延迟就这么诞生了。

解决: FastClick 是 FT Labs 专门为解决移动端浏览器 300 毫秒点击延迟问题所开发的一个轻量级的库。简而言之, FastClick 在检测到 touchend 事件的时候, 会通过 DOM 自定义事件立即触发一个模拟 click 事件的 click 事件 (自定义事件), 并把浏览器在 300 毫秒之后真正触发的 click 事件阻止掉。

8. 移动端的适配, rem+媒体查询/meta 头设置。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

首先我们拿到美工给的图一般会分为两种格式一种是 640 的图一种是 750 的图这两种图其实的效果是一样的只不过给我们的测量数据不同，所以其他都是一样的，这里我们就拿 640 的来讲解。

我们将屏幕分为 20 份（这里随便只不过 20 份比较好计算），然后将 640 的屏幕分为 20 份每一份是 32px;

所以我们在计算的时候例如：获得一个尺寸为 28 转化为 对应的比例关系就是 $28/32\text{ rem}$!

我们以前介绍过媒体查询，这里也要为了使用不同的尺寸来进行媒体查询的设置，只不过是为了兼容尺寸，可以把

它放到 base.css 文件中!

```
/* 媒体查询 字号 分为20份 */
@media screen and (width:320px){
    html{
        font-size:320/20px;
    }
}
@media screen and (width:360px){
    html{
        font-size:360/20px;
    }
}
@media screen and (width:375px){
    html{
        font-size:375/20px;
    }
}
@media screen and (width:414px){
    html{
        font-size:414/20px;
    }
}
@media screen and (width:412px){
    html{
        font-size:412/20px;
    }
}
```

然后就开始写布局了，这期间一定要记住，这里使用的是 rem，得到的尺寸都要除以 32，使用的单位是 rem!

```
<!-- 优先使用 IE 最新版本和 Chrome -->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>
<!-- 为移动设备添加 viewport -->
<meta name="viewport" content="width=device-width,initial-scale=1, maximum-scale=1,
minimum-scale=1, user-scalable=no">
<!-- 添加到主屏后的标题（iOS 6 新增） -->
<meta name="apple-mobile-web-app-title" content="">
<!-- 是否启用 WebApp 全屏模式，删除苹果默认的工具栏和菜单栏 -->
<meta name="apple-mobile-web-app-capable" content="yes"/>
<!-- 隐藏状态栏/
设置状态栏颜色：只有在开启WebApp全屏模式时才生效。content的值为default | black |
black-translucent -->
<meta name="apple-mobile-web-app-status-bar-style" content="black"/> <!--
添加到主屏后的标题 -->
<meta name="apple-mobile-web-app-title" content="标题">
<!-- 添加智能 App 广告条 Smart App Banner（iOS 6+ Safari） -->
<meta name="apple-itunes-app" content="app-id=myAppStoreID,
affiliate-data=myAffiliateData, app-argument=myURL">
<!-- 忽略页面中的数字识别为电话，忽略email识别 -->
<meta name="format-detection" content="telephone=no, email=no"/>
<!-- 下面三个是清除缓存 微信浏览器缓存严重又无刷新；这个方法调试的时候很方便--> <meta
http-equiv="Pragma" content="no-cache">
<meta http-equiv="Cache-Control" content="no-cache">
<meta http-equiv="Expires" content="0"> <title>document title</title>
```

9. 移动端的手势和事件。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

触摸事件

该类事件会在用户手指放在屏幕上时，在屏幕上滑动时，或从屏幕上移开时触发。具体来说有以下几个

触摸事件。

- Touchstart 当手指放在屏幕上触发。
- Touchmove 当手指在屏幕上滑动时，连续地触发。
- Touchend 当手指从屏幕上离开时触发。
- Touchcancel 当系统停止跟踪时触发，系统什么时候取消，文档没有明确的说明。

【总】以上四个，是 w3c 提供的触摸事件，只针对触摸设备，最常用的是前三个。

由于触摸会导致屏幕动来动去，所以可以在这些事件的事件处理函数内使用 `event.preventDefault()`，来阻止屏幕的默认滚动。

- 除了常用的 DOM 属性，触摸事件还包含下列三个用于跟踪触摸的属性。
- `touches`：表示当前跟踪的触摸操作的 touch 对象的数组。

当一个手指在触屏上时，`event.touches.length=1`，

当两个手指在触屏上时，`event.touches.length=2`，以此类推。

- `targetTouches`：特定于事件目标的 touch 对象数组。

因为 touch 事件是会冒泡的，所以利用这个属性指出目标对象。

- `changedTouches`：表示自上次触摸以来发生了什么改变的 touch 对象的数组。

每个 touch 对象都包含下列几个属性：

- `clientX`：触摸目标在视口中的 x 坐标。

`clientY`：触摸目标在视口中的 y 坐标。

`identifier`：标识触摸的唯一 ID。

`pageX`：触摸目标在页面中的 x 坐标。

`pageY`：触摸目标在页面中的 y 坐标。

`screenX`：触摸目标在屏幕中的 x 坐标。

`screenY`：触摸目标在屏幕中的 y 坐标。

`target`：触摸的 DOM 节点目标。

- 使用 `clientX`……时，必须要指明具体的 touch 对象，而不要直接指明数组。

`event.touches[0]`

在 `touchend` 事件处理函数中，当该事件发生时，`touches` 里面已经没有任何的 touch 对象了，此时，就要使用 `changeTouches` 集合。

手势事件

- gesturestart: 当一个手指已经按在屏幕上, 而另一个手指又触摸在屏幕时触发。
- gesturechange: 当触摸屏幕的任何一个手指的位置发生变化时触发。
- gestureend: 当任何一个手指从屏幕上面移开时触发。

【注意】只有两个手指都触摸到事件的接收容器时才触发这些手势事件。

- 触摸事件与手势事件之间的关系

1、当一个手指放在屏幕上时, 会触发 touchstart 事件, 如果另一个手指又放在了屏幕上, 则会触发 gesturestart 事件, 随后触发基于该手指的 touchstart 事件。

2、如果一个或两个手指在屏幕上滑动, 将会触发 gesturechange 事件, 但只要有一个手指移开, 则会触发 gestureend 事件, 紧接着又会触发 toucheend 事件。

- 手势的专有属性
- rotation: 表示手指变化引起的旋转角度, 负值表示逆时针, 正值表示顺时针, 从零开始。
- scale: 表示两个手指之间的距离情况, 向内收缩会缩短距离, 这个值从 1 开始, 并随距离拉大而增长。

10. unicode, utf8, gbk 编码的了解, 乱码的解决。

你可以这样回答: (提示: 回答该问题建议: 请理解内容, 使用自己的语言表达为最佳的回答哦!)

Unicode

1) .可以将 Unicode 编码理解为国际唯一标准编码, 中间编码, 最底层的编码, 它强大到可以编码这世界上所有的语言的所有文字。可以将它形象的比喻为中间人。unicode 普遍是用十六进制表示\u, 也可以用十进制, 二进制表示。其实, 就是二进制码, 双字节表示的二进制码。

2). utf-8 和 gbk 到底是什么?

UTF-8 (8-bit Unicode Transformation Format) 是一种针对 Unicode 的可变长度字符编码, 又称万国码。

gbk, utf-8 都是区域性编码(美国人制定的编码, 国际性编码), 他们都是由 Unicode 编码封装而成(再次编码)。

UTF8 是为传送 unicode 而想出来的“再编码”方法，url 传输以及其它传输中用的编码都是 utf8 编码。gbk 则是为了汉字而制定的编码(中华人民共和国全国信息技术标准化技术委员会 1995 年 12 月 1 日制订)。

3). 下面, 还是以汉字"严"为例, 演示如何实现 unicode 转换为 UTF-8 编码?

已知"严"的 unicode 是 4E25(1001110 00100101), 根据上表, 可以发现 4E25 处在第三行的范围内(0000 0800 - 0000 FFFF), 因此"严"的 UTF-8 编码需要三个字节, 即格式是

1110xxxx 10xxxxxx 10xxxxxx". 然后, 从"严"的最后一个二进制位开始, 依次从后向前填入格式中的 x, 多出的位补 0. 这样就得到了, "严"的 UTF-8 编码是 "11100100 10111000 10100101", 转换成十六进制就是 E4B8A5。

[illegible]

乱码的解决:


```
//转码
    str= new String(str.getBytes("iso8859-1"),"gb2312");
    str= new String(str.getBytes("iso8859-1"),"GBK");
//JSP中乱码
    <%@pagelanguage="java"contentType="text/html;charset=GBK" pageEncoding="GBK" %>
    tomcat中配置
        更改 Tomcat\conf\server.xml, 指定浏览器的编码格式为“简体中文”:
        方法是找到 server.xml 中的
    <Connector port="8080" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        connectionTimeout="20000" disableUploadTimeout="true" URIEncoding='GBK' />
// servlet中
    response.setContentType("text/html; charset=GBK");
```

14. 技术拓展加分（拔高）

1. 你都看过什么书？最近在看什么书。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

这个题的思路是：这个书要和前端相关。一定要记住书名，最好的真正的读过，知道这本书的核心内容是讲什么的。

2. 用过什么框架？有没有看过什么框架的代码。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

通过这些年的工作，我用过的框架和类库有 jQuery、bootstrap、zepto、vue、react 等前端流行的一些框架和类库。我看过 jQuery 的源码。自己也仿照 jQuery 的源码写过封装。

3. 有没有学过设计模式。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

设计模式是在软件工程实践过程中，程序员们总结出的良好的编程方法，使用设计模式能够增加系统

的健壮性，易修改性和可扩展性，当你进行开发的软件规模比较大的时候，良好的设计模式会给编程带来便利，让系统更加稳定，这些在自己编写小程序的时候是体现不出来的。现在大多数框架都使用了很多设计模式，正是因为有了这些设计模式，才能让程序更好的工作，例如烟水晶框架的单例模式，struts 的 mvc 模式，java 类库中 iterator 的迭代器模式等等，都是设计模式良好的应用。自己在写代码的时候，如果能合理的使用设计模式，一定能让你的面向对象编程大放光彩，在系统模块化和信息隐藏方面做的更好。至于怎么学好设计模式，一定是多练，多看，headfirst design pattern 和大话设计模式这两本书都不错，很通俗易懂，可以多多参考。

4. 说一说观察者模式。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

观察者模式（有时又被称为模型-视图（View）模式、源-收听者(Listener)模式或从属者模式）是软件设计模式的一种。在此种模式中，一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实现事件处理系统。

观察者模式（Observer）完美的将观察者和被观察的对象分离开。举个例子，用户界面可以作为一个观察者，业务数据是被观察者，用户界面观察业务数据的变化，发现数据变化后，就显示在界面上。面向对象设计的一个原则是：系统中的每个类将重点放在某一个功能上，而不是其他方面。一个对象只做一件事情，并且将他做好。观察者模式在模块之间划定了清晰的界限，提高了应用程序的可维护性和重用性。观察者设计模式定义了对象间的一种一对多的组合关系，以便一个对象的状态发生变化时，所有依赖于它的对象都得到通知并自动刷新。

观察者模式有很多实现方式，从根本上说，该模式必须包含两个角色：观察者和被观察对象。在刚才的例子中，业务数据是被观察对象，用户界面是观察者。观察者和被观察者之间存在“观察”的逻辑关联，

当被观察者发生改变的时候，观察者就会观察到这样的变化，并且做出相应的响应。如果在用户界面、业务数据之间使用这样的观察过程，可以确保界面和数据之间划清界限，假定应用程序的需求发生变化，需要修改界面的表现，只需要重新构建一个用户界面，业务数据不需要发生变化。

4. 你最大的优点是什么？那你最大的缺点呢？

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

优点：

沉着冷静、调理清楚、立场坚定、顽强向上、乐于助人和关心他人、适应能力和幽默感、乐观和友爱。我在前端这个行业也有几年工作经验，能迅速的融入团队为公司做出贡献

缺点：

被面试官问的概率很大，也是 HR 的杀手锏和狠招，这个问题最难回答，通常面试官不希望听到求职直接回答的缺点是什么，如果求职者说自己小心眼、脾气大、工作效率低，企业肯定不会录用你。不要自作聪明地回答“我最大的缺点就是过于追求完美”，有的人以为这样回答会显得自己比较出色，但事实上，他已经岌岌可危了。面试官喜欢求职者从自己的优点说起，中间加一些小缺点，最后再把问题转到优点上，突出优点的部分，面试官喜欢聪明的求职者。

回答范例：这个问题好难回答啊！我想想……（亲和力表现，也缓解了自己的紧张情绪）

我的缺点是比较执着，比如在技术方面比较爱钻研，有的时候会为一个技术问题加班到深夜。还有就是，工作比较按部就班，总是按照项目经管的要求完成任务。另外的缺点是，总在息的工作范围内有创新意识，并没有扩展给其他同事。这些问题我想我可以进入公司后以最短的时间来解决，我的学习能力很强，我相信可以很快融入公司的企业文化，进入工作状态。我想就这些吧。

5.平时看哪些博客

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

平时处理些一些播客意外有时候还会在知乎上写一点儿东西，我有个个人的微信公众号，有时间我也会在上面写一点自己的心得来分享我的技术，让跟多的人少走一地儿弯路。

6. 现在你的领导给你了一份工作，要求你一个星期完成，但你看了需求以后估计需要 3 周才能完成，你该怎么办？

- 1 总结分析工作的性质,从重要度与难度上对工作分类.
- 2 根据分类,把重要的工作放在前面做,次要的后面做.重要的工作要细致,尽量做到不出差错.
- 3 协调人力来替你分担.无论请客也好,还是命令下面小弟也好.总之,其他事情让别人帮你做,重要的事情自己把关.
- 4 全心投入,不要想太多了.赶紧著吧! 同时你也要事先讲好如果三个周期完成的要一周完成会有哪些风险如果领导同意的话你就加油吧。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

7. 平时关注的前端技术

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我这段时间在不断的学习 VUE，Vue.js 是一套构建用户界面的渐进式框架。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。

Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。

另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序

序提供驱动。

Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

优势：

1.简洁 2. 轻量 3.快速 4. 数据驱动 5.模块友好 6. 组件化

8. 如何规划自己的职业生涯

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

大部分面试官都会问你是否有职业规划，这个问题的背后是了解你的求职动机和对自己中长期职业发展的思考。在回答这个问题之前，要对自己有个清晰的认识，知道自己想往哪个方向发展以及未来有什么计划，要给面试官一种积极向上，好学上进，有追求，有规划的感觉，面试官喜欢有规划的求职者。

回答范例：我希望从现在开始，1-2年之内能够在我目前申请的这个职位上沉淀下来，通过不断的努力后，最好能有晋升，希望3-5年内可以从开发做到架构师。同时我也希望自己能够在企业的平台上得到进一步的职业能力提升。

9. 项目过程中，有遇到什么问题吗？怎么解决的

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

开发项目不可能没有问题，这就需要你针对你的熟悉的项目去准备几个问题，这个问题的最好是关于这个项目核心想要表达的技术的，当然了有了问题就一定要准备好答案。

10. 最近在研究哪方面的东西

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我这段时间在不断的学习 VUE，Vue.js 是一套构建用户界面的渐进式框架。与其他重量级框架不同的

是，Vue 采用自底向上增量开发的设计。

Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。

另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

优势：

1.简洁 2. 轻量 3.快速 4. 数据驱动 5.模块友好 6. 组件化

11. 请介绍一项你最热爱、最擅长的专业领域，并且介绍的学习规划

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

回答这个问题时面试官分两种情况，只是简单的问下大致了解下你的技术水平或是就你回答擅长方面进行提问。

对于第一种情况你可以这样回答：

向各种技术网站学习（我比较喜欢的是思、掘金等）网站细致学，html,css,html5,css3,javascript,jquery.边看边练习上面的例子。

每一个部分学习完后，找一个文本工具，把学到的知识回忆并写下来，这样的学习效果不错。

循序渐进，先学并牢记知识结构和每个结构涉及的常用知识，对于每个结构详细的知识可以在使用时再来参考，在用中学慢慢就熟悉了。

提高代码质量，更深入学习语言 选择看两本讲代码优化，性能优化方法的书籍，我选择 js 语言精粹、精通 js。（对于初级的来说可以看犀牛书和红皮书）。

但是对于第二中得话往往出现在你的面试情况不太理想的情况下面试官觉得从你的简历上问你的问题你回答的不是很理想，但是面试官觉得你还不错想继续聊下。所以对于你擅长的这个问题要慎重回答。千万不要给自己挖坑。

12. 请介绍你参与的印象最深刻的一个项目，为什么？并且介绍你在项目中的角色和发挥的作用。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

举一个你最有把握的例子，把来龙去脉说清楚，而不要说了很多却没有重点。切忌夸大其词，把别人的功劳到说成自己的，很多主管为了确保要用的人是最适合的，会打电话向你的前一个主管征询对你的看法及意见，所以如果说谎，是很容易穿梆的。同时这个问题在技术面试时常被问到，问这个问题的意图是想考察你的成长路径和编程习惯，因为，你最熟悉的项目往往是你成长最快的项目，那个成长最快的项目往往会给你今后的编程习惯留下很多痕迹。所以，通过你对熟悉项目的描述，有经验的他会很快锁定你技术成长中的缺陷和闪光点，从而判断是否能够“为我所用”。

你最好拿出一个自己最擅长技术的那个项目进行介绍，他听完你的介绍后，会接下来进行提问，这样他所有问的问题，你都成竹在胸了。

切忌拿自己参与很少的项目来介绍，一旦他深入的询问很可能你会答非所问，反而造成更严重的影响。你大可以和他谈谈在那个项目中获得的经验，这样会引起此君的共鸣，有可能的话，说出一些你自己的小技巧，他会很高兴，同时这场面试也会很轻松，拿到 Offer 基本没问题了。

15. HR 面试的开放性问题（体现积极向上的人生观）

1.你为什么要学习前端。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

前端开发入门门槛低,但需求旺盛,特别是有经验的前端开发人员。

前端开发短期可以积累足够的经验,比如 3 年,而后端开发想要积累同等程度的经验至少需要 5 年。

前端开发技术变化慢,HTML5 定稿都需要 8 年!而后端开发技术更新很快,各种框架,架构模式变更迅速,需要时刻学习。

前端开发技术发展越来越成熟,且适用范围更广。比如 HTML5 可以替代原生 APP(性能方面有待考

究),JavaScript 能够用于数据库操作(MongoDB 等 NoSQL 支持 JS 语法),NodeJS 能够让 JavaScript 在服务器端运行,只要会 Js 语法不用后端开发语言照样开发服务端程序。

前端开发技术支持全栈式开发,不需要后端开发语言的支持。

由于互联网行业的极速扩张,各大公司对前端工程师的需求非常旺盛,自然钱景旺旺。加上 HTML5 规范的最终定稿,必定引起 web 的热潮。NodeJS 在服务器端的延伸也使 JS 大放光彩

2. 你平时的是怎么学习前端的？有什么输出。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我都是同过自学的方式来学习前端知识，会买一写相关的书籍来学习，每一个技术学习完毕我会做个比较综合的案例来巩固我的所学，当然我会在学习每个小的知识点的时候也会做一些小的案例。

3.你觉得自己最好的项目是什么。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

最好说你简历上的项目这样你比较了解。

4. 身边比较佩服的人有什么值得你学习的？你为什么没有跟他们一样。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我比较佩服我之前的一个领导，他工作认真扶着，能力出众，关键是他能很好的处理同事之间的关系以及她的工作和生活中的事情。他的脸上经常能看到微笑。我现在正在努力的成长中，想尽快成为想他一样的人。

5. 同事的什么问题会让你接受不了

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我觉得是带着情绪来上班吧。我个人觉得生活中的事情不应该带到工作中，不过我遇了我会和他（她）聊

聊，开导他（她）一下，避免一天的心情不好影响工作。

6. 压力最大的事情是什么

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

你可以从不同方面的压力去思考,权衡一下,说什么最好.如工作中的最大压力是什么,如何解决的,如控制情绪、各种解决措施等.生活中的压力如就业,收入等,人际交往等.但权衡一下,看他们问的意向,你多准备几种情况,相机行事.

压力可定会有，但是我喜欢接受新的工作挑战。当然来到一个新的环境中从事新的项目工作，也许技术上有可能比较缺少贵公司技术的经验，所以需要缜密的思考和妥善的规划。我的工作压力多产生在新工作的计划和初级实施阶段，因为这时候总是会产生很多新的问题。这样就需要我不断学习新的知识，来应对工作压力，直到全部工作顺利完成。虽然刚开始有压力，但那份成就感却也是让人满足而且值得回味的。

7. 身边的朋友通常对你的评价是什么

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我的朋友都说我是一个可以信赖的人。因为，我一旦答应别人的事情，就一定会做到。如果我做不到，我就不会轻易许诺。

我觉的我是一个比较随和的人，与不同的人都可以友好相处。在我与人相处时，我总是能站在别人的角度考虑问题。

8. 喜欢什么样的工作氛围

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我理想中的公司氛围是公司像一个家庭一样，能够带给我像家一样的享受。

正式出来工作，进入了一个大的家庭，在这个家庭里面有矛盾，有不满，但是更多的是彼此和平相处。不能说它完全符合我心目中的团队形象，但是在某种程度上也基本上符合了我心目中团队形象。

在这个大家庭里面有一个坚实的领导团队，领导团队里面的领导各司其职，也能够做到管理好下属，准确地把握公司发展方向，能够高高在上让人敬畏，也能够接地气和职员和谐相处。除了领导团队之外，公司的员工也是在做好自己工作的基础上互相帮助，涉及到团队的事情有商有量，涉及到工作能够互相指导，全公司上下工作和生活都是井然有序的，使得每个职员都是非常规律的在做着自己的工作。

除了这种和谐相处之外，其实我更多的是希望公司的职员，能够做到像朋友一样。公司的职员不是只做到在上班的时候互相打个招呼互相帮助，在私底下也要有所联系。不仅仅是在工作中交流，在私底下我们也能够做到像朋友一样多多交流，甚至是偶尔逛逛街吃吃饭这样。

其实我羡慕的一种公司氛围更多的就是说公司像一个家庭一样，能够带给我像家一样的享受。

8. 如何看待加班

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

如果是工作需要我会义不容辞加班，我现在单身，没有任何家庭负担，可以全身心的投入工作。但同时，我也会提高工作效率，减少不必要的加班。

9. 如何看待加班

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

没有：我现在还是单身，现在的生活节奏很快，我想先以工作为主，等工作事业稳定了以后再考虑这方面的事情。周末有时候我会和朋友一起出去打打球，旅游也很喜欢。有时候也看看技术方面的播客。

有：我有对象了，但是现在还没有结婚的计划，我们计划的先奋斗几年，有了一定的积蓄以后再考虑结婚的事情。平时周末天气合适的话会出去玩玩，有时候我们会一起做点儿东西，或者各自看看各自喜欢的书

10. 未来职业规划

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

我希望从现在开始，1-2年之内能够在我目前申请的这个职位上沉淀下来，通过不断的努力后，最好能有晋升，希望3-5年内可以从开发做到架构师。同时我也希望自己能够在企业的平台上得到进一步的职业能

力提升。

11. 未来职业规划

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

北京，我现在已经收到 offer 了。根据我对贵公司的了解，以及我在这份工作上所累积的专业、经验及人脉，相信正是贵公司所找寻的人才。而我在工作态度、EQ 上，也有圆融、成熟的一面，和主管、同事都能合作愉快

11. 你现在手里有没有 offer。

你可以这样回答：（提示：回答该问题建议：请理解内容，使用自己的语言表达为最佳的回答哦！）

关于这个问题的话如果前面技术了得还不错的话，不管你有没有你都可以说有一家但是不太满意（可以从距离、公司环境等）不如您的公司的职位的匹配度高。