面试宝典

一、HTML 和 CSS

1. 你做的页面在哪些流览器测试过？这些浏览器的内核分别是什么？

IE: trident 内核

Firefox：gecko 内核

Safari:webkit 内核

Opera:以前是 presto 内核，Opera 现已改用 Google Chrome 的 Blink 内核

Chrome:Blink(基于 webkit，Google 与 Opera Software 共同开发)

2. 每个 HTML 文件里开头都有个很重要的东西，Doctype，知道这是干什么的吗?

<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。（重点：告诉浏览器按照何种规范解析页面）

3. Quirks 模式是什么？它和 Standards 模式有什么区别

从 IE6 开始，引入了 Standards 模式，标准模式中，浏览器尝试给符合标准的文档在规范上的正确处理达到在指定浏览器中的程度。

在 IE6 之前 CSS 还不够成熟，所以 IE5 等之前的浏览器对 CSS 的支持很差， IE6 将对 CSS 提供更好的支持，然而这时的问题就来了，因为有很多页面是基于旧的布局方式写的，而如果 IE6 支持 CSS 则将令这些页面显示不正常，如何在即保证不破坏现有页面，又提供新的渲染机制呢?

在写程序时我们也会经常遇到这样的问题，如何保证原来的接口不变，又提供更强大的功能，尤其是新功能不兼容旧功能时。遇到这种问题时的一个常见做法是增加参数和分支，即当某个参数为真时，我们就使用新功能，而如果这个参数 不为真时，就使用旧功能，这样就能不破坏原有的程序，又提供新功能。IE6 也是类似这样做的，它将 DTD 当成了这个“参数”，因为以前的页面大家都不会去写 DTD，所以 IE6 就假定 如果写了 DTD，就意

味着这个页面将采用对 CSS 支持更好的布局，而如果没有，则采用兼容之前的布局方式。这就是 Quirks 模式（怪癖模式，诡异模式，怪异模式）。

区别：

总体会有布局、样式解析和脚本执行三个方面的区别。

盒模型：在 W3C 标准中，如果设置一个元素的宽度和高度，指的是元素内容的宽度和高度，而在 Quirks 模式下，IE 的宽度和高度还包含了 padding 和 border。

设置行内元素的高宽：在 Standards 模式下，给等行内元素设置 wdith 和 height 都不会生效，而在 quirks 模式下，则会生效。

设置百分比的高度：在 standards 模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置百分比的高度，子元素设置一个百分比的高度是无效的用 margin:0 auto 设置水平居中：使用 margin:0 auto 在 standards 模式下可以使元素水平居中，但在 quirks 模式下却会失效。

（还有很多，答出什么不重要，关键是看他答出的这些是不是自己经验遇到的，还是说都是看文章看的，甚至完全不知道。）

4. div+css 的布局较 table 布局有什么优点?

改版的时候更方便 只要改 css 文件。

页面加载速度更快、结构化清晰、页面显示简洁。

表现与结构相分离。

易于优化（seo）搜索引擎更友好，排名更容易靠前。

5. img 的 alt 与 title 有何异同? strong 与 em 的异同?

a:alt(alt text):为不能显示图像、窗体或 applets 的用户代理（UA），alt 属性用来指定替换文字。替换文字的语言由 lang 属性指定。(在 IE 浏览器下会在没有 title 时把 alt 当成 tool tip 显示)

title(tool tip):该属性为设置该属性的元素提供建议性的信息。

strong:粗体强调标签，强调，表示内容的重要性

em:斜体强调标签，更强烈强调，表示内容的强调点

6. 你能描述一下渐进增强和优雅降级之间的不同吗?

渐进增强 progressive enhancement：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级 graceful degradation：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

“优雅降级”观点
“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。而将那些被认为“过时”或有功能缺失的浏览器下的测试工作安排在开发周期的最后阶段，并把测试对象限定为主流浏览器（如 IE、Mozilla 等）的前一个版本。

在这种设计范例下，旧版的浏览器被认为仅能提供“简陋却无
妨 (poor, but passable)” 的浏览体验。你可以做一些小的调整来适应某个特定的浏览器。但由于它们并非我们所关注的焦点，因此除了修复较大的错误之外，其它的差异将被直接忽略。

“渐进增强”观点
“渐进增强”观点则认为应关注于内容本身。

内容是我们建立网站的诱因。有的网站展示它，有的则收集它，有的寻求，有的操作，还有的网站甚至会包含以上的种种，但相同点是它们全都涉及到内容。这使得“渐进增强”成为一种更为合理的设计范例。这也是它立即被 Yahoo! 所采纳并用以构建其“分级式浏览器支持 (Graded Browser Support)”策略的原因所在。

那么问题来了。现在产品经理看到 IE6,7,8 网页效果相对高版本现代浏览器少了很多圆角，阴影（CSS3），要求兼容（使用图片背景，放弃 CSS3），你会如何说服他?

7. 为什么利用多个域名来存储网站资源会更有效?

CDN 缓存更方便

突破浏览器并发限制

节约 cookie 带宽

节约主域名的连接数，优化页面响应速度

防止不必要的安全问题

8. 请谈一下你对网页标准和标准制定机构重要性的理解。

网页标准和标准制定机构都是为了能让 web 发展的更‘健康’，开发者遵循统一的标准，降低开发难度，开发成本，SEO 也会更好做，也不会因为滥用代码导致各种 BUG、安全问题，最终提高网站易用性。

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

sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

web storage 和 cookie 的区别

Web Storage 的概念和 cookie 相似，区别是它是为了更大容量存储设计的。Cookie 的大小是受限的，并且每次你请求一个新的页面的时候 Cookie 都会被发送过去，这样无形中浪费了带宽，另外 cookie 还需要指定作用域，不可以跨域调用。

除此之外，Web Storage 拥有 setItem,getItem,removeItem,clear 等方法，不像 cookie 需要前端开发者自己封装 setCookie，getCookie。但是 Cookie 也是不可以或缺的：Cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在 ，而 Web Storage 仅仅是为了在本地“存储”数据而生。

10. 简述一下 src 与 href 的区别。

src 用于替换当前元素，href 用于在当前文档和引用资源之间确立联系。

src 是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素。

<script src =”js.js”></script>

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将 js 脚本放在底部而不是头部。

href 是 Hypertext Reference 的缩写，指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加

<link href=”common.css” rel=”stylesheet”/>

那么浏览器会识别该文档为 css 文件，就会并行下载资源并且不会停止对当前文档的处理。这也是为什么建议使用 link 方式来加载 css，而不是使用@import 方式。

11. 知道的网页制作会用到的图片格式有哪些?

png-8，png-24，jpeg，gif，svg。

但是上面的那些都不是面试官想要的最后答案。面试官希望听到是 Webp。（是否有关注新技术，新鲜事物）
科普一下 Webp：WebP 格式，谷歌（google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有 JPEG 的 2/3，并能节省大量的服务器带宽资源和数据空间。Facebook Ebay 等知名网站已经开始测试并使用 WebP 格式。
在质量相同的情况下，WebP 格式图像的体积要比 JPEG 格式图像小 40%

12. 知道什么是微格式吗？谈谈理解。在前端构建中应该考虑微格式吗?

微格式（Microformats）是一种让机器可读的语义化 XHTML 词汇的集合，是结构化数据的开放标准。是为特殊应用而制定的特殊格式。

优点：将智能数据添加到网页上，让网站内容在搜索引擎结果界面可以显示额外的提示。（应用范例：豆瓣，有兴趣自行 google）

13. 在 css/js 代码上线之后开发人员经常会优化性能，从用户刷新网页开始，一次 js 请求一般情况下有哪些地方会有缓存处理?

答案：dns 缓存，cdn 缓存，浏览器缓存，服务器缓存。

14. 一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为 css 图片，可以使用 CSSsprite，SVGsprite，Iconfont、Base64 等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

15. 你如何理解 HTML 结构的语义化?

去掉或样式丢失的时候能让页面呈现清晰的结构：

html 本身是没有表现的，我们看到例如<h1>是粗体，字体大小 2em，加粗；是加粗的，不要认为这是 html 的表现，这些其实 html 默认的 css 样式在起作用，所以去掉或样式丢失的时候能让页面呈现清晰的结构不是语义化的 HTML 结构的优点，但是浏览器都有有默认样式，默认样式的目的也是为了更好的表达 html 的语义，可以说浏览器的默认样式和语义化的 HTML 结构是不可分割的。

屏幕阅读器（如果访客有视障）会完全根据你的标记来“读”你的网页.

例如,如果你使用的含语义的标记,屏幕阅读器就会“逐个拼出”你的单词,而不是试着去对它完整发音.

PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对 CSS 的支持较弱）

使用语义标记可以确保这些设备以一种有意义的方式来渲染网页.理想情况下,观看设备的任务是符合设备本身的条件来渲染网页.

语义标记为设备提供了所需的相关信息,就省去了你自己去考虑所有可能的显示情况（包括现有的或者将来新的设备）.例如,一部手机可以选择使一段标记了标题的文字以粗体显示.而掌上电脑可能会以比较大的字体来显示.无论哪种方式一旦你对文本标记为标题,您就可以确信读取设备将根据其自身的条件来合适地显示页面.

搜索引擎的爬虫也依赖于标记来确定上下文和各个关键字的权重

过去你可能还没有考虑搜索引擎的爬虫也是网站的“访客”,但现在它们他们实际上是极其宝贵的用户.没有他们的话,搜索引擎将无法索引你的网站,然后一般用户将很难过来访问.

你的页面是否对爬虫容易理解非常重要,因为爬虫很大程度上会忽略用于表现的标记,而只注重语义标记.

因此,如果页面文件的标题被标记,而不是,那么这个页面在搜索结果的位置可能会比较靠后.除了提升易用性外,语义标记有利于正确使用 CSS 和 JavaScript,因为其本身提供了许多“钩钩”来应用页面的样式与行为.

SEO 主要还是靠你网站的内容和外部链接的。

便于团队开发和维护

W3C 给我们定了一个很好的标准，在团队中大家都遵循这个标准，可以减少很多差异化的东西，方便开发和维护，提高开发效率，甚至实现模块化开发。

16. 谈谈以前端角度出发做好 SEO 需要考虑什么?

了解搜索引擎如何抓取网页和如何索引网页

你需要知道一些搜索引擎的基本工作原理，各个搜索引擎之间的区别，搜索机器人（SE robot 或叫 web crawler）如何进行工作，搜索引擎如何对搜索结果进行排序等等。

Meta 标签优化

主要包括主题（Title)，网站描述(Description)，和关键词（Keywords）。还有一些其它的隐藏文字比如 Author（作者），Category（目录），Language（编码语种）等。

如何选取关键词并在网页中放置关键词

搜索就得用关键词。关键词分析和选择是 SEO 最重要的工作之一。首先要给网站确定主关键词（一般在 5 个上下），然后针对这些关键词进行优化，包括关键词密度（Density），相关度（Relavancy），突出性（Prominency）等等。

了解主要的搜索引擎

虽然搜索引擎有很多，但是对网站流量起决定作用的就那么几个。比如英文的主要有 Google，Yahoo，Bing 等；中文的有百度，搜狗，有道等。不同的搜索引擎对页面的抓取和索引、排序的规则都不一样。还要了解各搜索门户和搜索引擎之间的关系，比如 AOL 网页搜索用的是 Google 的搜索技术，MSN 用的是 Bing 的技术。

主要的互联网目录

Open Directory 自身不是搜索引擎，而是一个大型的网站目录，他和搜索引擎的主要区别是网站内容的收集方式不同。目录是人工编辑的，主要收录网站主页；搜索引擎是自动收集的，除了主页外还抓取大量的内容页面。

按点击付费的搜索引擎

搜索引擎也需要生存，随着互联网商务的越来越成熟，收费的搜索引擎也开始大行其道。最典型的有 Overture 和百度，当然也包括 Google 的广告项目 Google Adwords。越来越多的人通过搜索引擎的点击广告来定位商业网站，这里面也大有优化和排名的学问，你得学会用最少的广告投入获得最多的点击。

搜索引擎登录

网站做完了以后，别躺在那里等着客人从天而降。要让别人找到你，最简单的办法就是将网站提交（submit）到搜索引擎。如果你的是商业网站，主要的搜索引擎和目录都会要求你付费来获得收录（比如 Yahoo 要 299 美元），但是好消息是（至少到目前为止）最大的搜索引擎 Google 目前还是免费，而且它主宰着 60%以上的搜索市场。

链接交换和链接广泛度（Link Popularity）

网页内容都是以超文本（Hypertext）的方式来互相链接的，网站之间也是如此。除了搜索引擎以外，人们也每天通过不同网站之间的链接来 Surfing（“冲浪”）。其它网站到你的网站的链接越多，你也就会获得更多的访问量。更重要的是，你的网站的外部链接数越多，会被搜索引擎认为它的重要性越大，从而给你更高的排名。

合理的标签使用

17. 有哪项方式可以对一个 DOM 设置它的 CSS 样式?

外部样式表，引入一个外部 css 文件

内部样式表，将 css 代码放在 <head> 标签内部

内联样式，将 css 样式直接定义在 HTML 元素内部

18. CSS 都有哪些选择器?

派生选择器（用 HTML 标签申明）

id 选择器（用 DOM 的 ID 申明）

类选择器（用一个样式类名申明）

属性选择器（用 DOM 的属性申明，属于 CSS2，IE6 不支持，不常用，不知道就算了）

除了前 3 种基本选择器，还有一些扩展选择器，包括

后代选择器（利用空格间隔，比如 div .a{ }）

群组选择器（利用逗号间隔，比如 p,div,#a{ }）

那么问题来了，CSS 选择器的优先级是怎么样定义的?
基本原则：

一般而言，选择器越特殊，它的优先级越高。也就是选择器指向的越准确，它的优先级就越高。

复杂的计算方法：

用 1 表示派生选择器的优先级

用 10 表示类选择器的优先级

用 100 标示 ID 选择器的优先级

div.test1 .span var 优先级 1+10 +10 +1

span#xxx .songs li 优先级 1+100 + 10 + 1

#xxx li 优先级 100 +1

那么问题来了，看下列代码，<p>标签内的文字是什么颜色的?

```
<style>
.classA{ color:blue;}
.classB{ color:red;}
</style>
<body>
<p class='classB classA'> 123 </p>
</body>
```

答案：red。与样式定义在文件中的先后顺序有关，即是后面的覆盖前面的，与在<p class=’ classB classA’ >中的先后关系无关。

19. CSS 中可以通过哪些属性定义，使得一个 DOM 元素不显示在浏览器可视范围内?

最基本的：

设置 display 属性为 none，或者设置 visibility 属性为 hidden

技巧性：

设置宽高为 0，设置透明度为 0，设置 z-index 位置在-1000em

20. 超链接访问过后 hover 样式就不出现的问题是什么？如何解决?

答案：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（link,visited,hover,active）

21. 什么是 Css Hack？ie6,7,8 的 hack 分别是什么?

答案：针对不同的浏览器写不同的 CSS code 的过程，就是 CSS hack。

示例如下：

```
#test{
    width:300px;
    height:300px;
    background-color:blue;      /*firefox*/
    background-color:red\9;      /*all ie*/
    background-color:yellow;    /*ie8*/
    +background-color:pink;       /*ie7*/
    _background-color:orange;       /*ie6*/    }
    :root #test { background-color:purple\9; }  /*ie9*/
@media all and (min-width:0px)
    { #test {background-color:black;} }  /*opera*/
@media screen and (-webkit-min-device-pixel-ratio:0)

{ #test {background-color:gray;} }       /*chrome and safari*/
```

22. 行内元素和块级元素的具体区别是什么？行内元素的 padding 和 margin 可设置吗?

块级元素(block)特性：

总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;

宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;

内联元素(inline)特性:

和相邻的内联元素在同一行;

宽度(width)、高度(height)、内边距的 top/bottom(padding-top/padding-bottom)和外边距的 top/bottom(margin-top/margin-bottom)都不可改变（也就是 padding 和 margin 的 left 和 right 是可以设置的），就是里面文字或图片的大小。
那么问题来了，浏览器还有默认的天生 inline-block 元素（拥有内在尺寸，可设置高宽，但不会自动换行），有哪些?
答案：<input> 、<img> 、<button> 、<texterea> 、<label>。

23. 什么是外边距重叠？重叠的结果是什么?

外边距重叠就是 margin-collapse。

在 CSS 当中，相邻的两个盒子（可能是兄弟关系也可能是祖先关系）的外边距可以结合成一个单独的外边距。这种合并外边距的方式被称为折叠，并且因而所结合成的外边距称为折叠外边距。

折叠结果遵循下列计算规则:

两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。

两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。

两个外边距一正一负时，折叠结果是两者的相加的和。

24. rgba()和 opacity 的透明效果有什么不同?

rgba()和 opacity 都能实现透明效果，但最大的不同是 opacity 作用于元素，以及元素内的所有内容的透明度，

而 rgba()只作用于元素的颜色或其背景色。（设置 rgba 透明的元素的子元素不会继承透明效果！）

25. css 中可以让文字在垂直和水平方向上重叠的两个属性是什么?

垂直方向：line-height

水平方向：letter-spacing

那么问题来了，关于 letter-spacing 的妙用知道有哪些么?
答案:可以用于消除 inline-block 元素间的换行符空格间隙问题。

26. 如何垂直居中一个浮动元素?

```
1
2     // 方法一：已知元素的高宽
3     #div1{
4        background-color:#6699FF;
5        width:200px;
6        height:200px;
7        position: absolute;      //父元素需要相对定位
8        top: 50%;
9        left: 50%;
10       margin-top:-100px ;  //二分之一的 height，width
11       margin-left: -100px;
12       }
13
14    //方法二:未知元素的高宽
15
16     #div1{
17       width: 200px;
18       height: 200px;
19       background-color: #6699FF;
20
21       margin:auto;
22       position: absolute;      //父元素需要相对定位
23       left: 0;
24       top: 0;
25       right: 0;
26       bottom: 0;
27       }
28
```

27.

那么问题来了，如何垂直居中一个<img>？（用更简便的方法。）

```
#container    //<img>的容器设置如下
{
    display:table-cell;
    text-align:center;
    vertical-align:middle;
}
```

27. px 和 em 的区别。

px 和 em 都是长度单位，区别是，px 的值是固定的，指定是多少就是多少，计算比较容易。em 得值不是固定的，并且 em 会继承父级元素的字体大小。

浏览器的默认字体高都是 16px。所以未经调整的浏览器都符合: 1em=16px。那么 12px=0.75em, 10px=0.625em。

28. 描述一个”reset”的 CSS 文件并如何使用它。知道 normalize.css 吗？你了解他们的不同之处?

重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置 CSS 文件并知道如何使用它们。他们是盲目的在做还是知道为什么这么做呢？原因是不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，或者更有戏剧性的性发生。

你可能会用 Normalize 来代替你的重置样式文件。它没有重置所有的样式风格，但仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，但又不扰乱其他的东西（如粗体的标题）。
在这一方面，无法做每一个复位重置。它也确实有些超过一个重置，它处理了你永远都不用考虑的怪癖，像 HTML 的 audio 元素不一致或 line-height 不一致。

29. Sass、LESS 是什么？大家为什么要使用他们?

他们是 CSS 预处理器。他是 CSS 上的一种抽象层。他们是一种特殊的语法/语言编译成 CSS。

例如 Less 是一种动态样式语言. 将 CSS 赋予了动态语言的特性，如变量，继承，运算，函数. LESS 既可以在客户端上运行 (支持 IE 6+, Webkit, Firefox)，也可一在服务端运行 (借助 Node.js)。
为什么要使用它们?

结构清晰，便于扩展。

可以方便地屏蔽浏览器私有语法差异。这个不用多说，封装对浏览器语法差异的重复处理，减少无意义的机械劳动。

可以轻松实现多重继承。

完全兼容 CSS 代码，可以方便地应用到老项目中。LESS 只是在 CSS 语法上做了扩展，所以老的 CSS 代码也可以与 LESS 代码一同编译。

30. display:none 与 visibility:hidden 的区别是什么?

display : 隐藏对应的元素但不挤占该元素原来的空间。
visibility: 隐藏对应的元素并且挤占该元素原来的空间。
即是，使用 CSS display:none 属性后，HTML 元素（对象）的宽度、高度等各种属性值都将“丢失”;而使用 visibility:hidden 属性后，HTML 元素（对象）仅仅是在视觉上看不见（完全透明），而它所占据的空间位置仍然存在。

31. CSS 中 link 和@import 的区别是：

Link 属于 html 标签，而@import 是 CSS 中提供的

在页面加载的时候，link 会同时被加载，而@import 引用的 CSS 会在页面加载完成后才会加载引用的 CSS

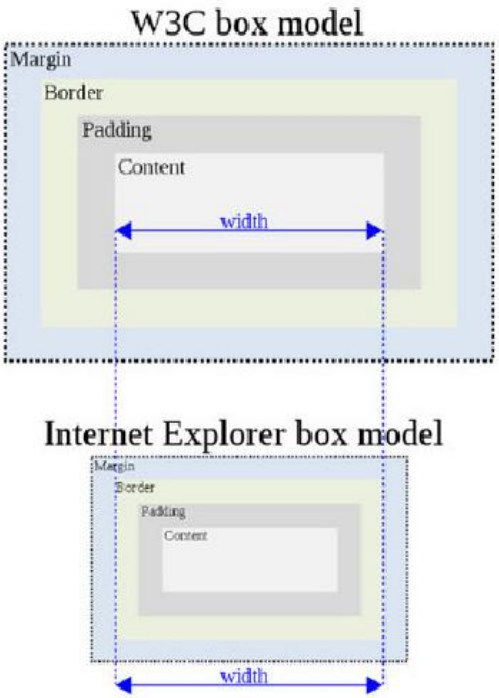
@import 只有在 ie5 以上才可以被识别，而 link 是 html 标签，不存在浏览器兼容性问题

Link 引入样式的权重大于@import 的引用（@import 是将引用的样式导入到当前的页面中)

32. 简介盒子模型：

CSS 的盒子模型有两种：IE 盒子模型、标准的 W3C 盒子模型模型

盒模型：内容、内边距、外边距（一般不计入盒子实际宽度）、边框



W3C和Internet Explorer的盒模型解释 width 属性的差异。

33. 为什么要初始化样式?

由于浏览器兼容的问题，不同的浏览器对标签的默认样式值不同，若不初始化会造成不同浏览器之间的显示差异

但是初始化 CSS 会对搜索引擎优化造成小影响

34. BFC 是什么?

BFC（块级格式化上下文），一个创建了新的 BFC 的盒子是独立布局的，盒子内元素的布局不会影响盒子外面的元素。在同一个 BFC 中的两个相邻的盒子在垂直方向发生 margin 重叠的问题

BFC 是指浏览器中创建了一个独立的渲染区域，该区域内所有元素的布局不会影响到区域外元素的布局，这个渲染区域只对块级元素起作用

35. html 语义化是什么?

当页面样式加载失败的时候能够让页面呈现出清晰的结构

有利于 seo 优化，利于被搜索引擎收录（更便于搜索引擎的爬虫程序来识别）

便于项目的开发及维护，使 html 代码更具有可读性，便于其他设备解析。

36. Doctype 的作用？严格模式与混杂模式的区别?

<!DOCTYPE>用于告知浏览器该以何种模式来渲染文档

严格模式下：页面排版及 JS 解析是以该浏览器支持的最高标准来执行

混杂模式：不严格按照标准执行，主要用来兼容旧的浏览器，向后兼容

37. IE 的双边距 BUG：块级元素 float 后设置横向 margin，ie6 显示的 margin 比设置的较大。

解决：加入_display：inline

38. HTML 与 XHTML——二者有什么区别?

1. 所有的标记都必须要有一个相应的结束标记
2. 所有标签的元素和属性的名字都必须使用小写
3. 所有的 XML 标记都必须合理嵌套
4. 所有的属性必须用引号 "" 括起来
5. 把所有 < 和 & 特殊符号用编码表示
6. 给所有属性赋一个值
7. 不要在注释内容中使用 "--"
8. 图片必须有说明文字

39. html 常见兼容性问题?

1.双边距 BUG float 引起的 使用 display

2.3 像素问题 使用 float 引起的 使用 dislpay:inline -3px

3.超链接 hover 点击后失效 使用正确的书写顺序 link visited hover active
4.Ie z-index 问题 给父级添加 position:relative
5.Png 透明 使用 js 代码 改
6.Min-height 最小高度！ Important 解决’
7.select 在 ie6 下遮盖 使用 iframe 嵌套
8.为什么没有办法定义 1px 左右的宽度容器（IE6 默认的行高造成的，使用 over:hidden,zoom:0.08 line-height:1px)
9.IE5-8 不支持 opacity，解决办法：

```
.opacity {
    opacity: 0.4
    filter: alpha(opacity=60); /* for IE5-7 */
    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; /* for IE 8*/
}
```

10. IE6 不支持 PNG 透明背景，解决办法: IE6 下使用 gif 图片

40. 对 WEB 标准以及 W3C 的理解与认识

答：标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外 链 css 和 js 脚本、结构行为表现的分离、文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，容易维 护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性。

41. 行内元素有哪些?块级元素有哪些?CSS 的盒模型?

答：块级元素：div p h1 h2 h3 h4 form ul
行内元素：a b br i span input select
Css 盒模型:内容，border ,margin，padding

42. 前端页面有哪三层构成，分别是什么?作用是什么?

答：结构层 Html 表示层 CSS 行为层 js。

43. Doctype 作用? 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?

(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。

44. 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：

<hr><img><input><link><meta>鲜为人知的是：

<area><base><col><command>

<embed><keygen><param><source><track><wbr>

45. CSS 的盒子模型?

（1）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).

46. 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?

* 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： font-size font-family color, UL LI DL DD DT;
* 不可继承： border padding margin width height ;
* 优先级就近原则，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。

47. 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%;
}
```

48. 浏览器的内核分别是什么?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？

* IE 浏览器的内核 Trident、 Mozilla 的 Gecko、google 的 WebKit、Opera 内核 Presto；

* png24 为的图片在 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.

* （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 {}

49. 列出 display 的值，说明他们的作用。position 的值， relative 和 absolute 定位原点是?

1. block 象块类型元素一样显示。

none 缺省值。向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。

2. position 的值

*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)。

* inherit 规定从父元素继承 position 属性的值。

50. absolute 的 containing block 计算方式跟正常流有什么不同?

lock-level boxes
一个 block-level element ('display' 属性值为 'block', 'list-item' 或是 ‘table’) 会生成一个 block-level box，这样的盒子会参与到 block-formatting context (一种布局的方式) 中。
block formatting context
在这种布局方式下，盒子们自所在的 containing block 顶部起一个接一个垂直排列，水平方向上撑满整个宽度 (除非内部的盒子自己内部建立了新的 BFC)。
containing block
一般来说，盒子本身就为其子孙建立了 containing block，用来计算内部盒子的位置、大小，而对内部的盒子，具体采用哪个 containing block 来计算，需要分情况来讨论：

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；
否则则由这个祖先元素的 padding box 构成。
根元素所在的 containing block 被称为 initial containing block，在我们常用的浏览器环境下，指的是原点与 canvas 重合，大小和 viewport 相同的矩形；
对于 position 为 static 或 relative 的元素，其 containing block 为祖先元素中最近的 block container box 的 content box (除 margin, border, padding 外的区域)；
对于 position:fixed 的元素，其 containing block 由 viewport 建立；
对于 position:absolute 的元素，则是先找到其祖先元素中最近的 position 属性非 static 的元素，然后判断：
如果都找不到，则为 initial containing block。

51. 对 WEB 标准以及 W3C 的理解与认识

标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外 链 css 和 js 脚本、结构行为表现的分离、文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，容易维 护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性；

52. css 的基本语句构成是?

选择器{属性 1:值 1;属性 2:值 2;……}

53. 浏览器标准模式和怪异模式之间的区别是什么?

盒子模型 渲染模式的不同
使用 window.top.document.compatMode 可显示为什么模式

54. CSS 中可以通过哪些属性定义，使得一个 DOM 元素不显示在浏览器可视范围内?

最基本的：

设置 display 属性为 none，或者设置 visibility 属性为 hidden

技巧性：

设置宽高为 0，设置透明度为 0，设置 z-index 位置在-1000

55. 行内元素和块级元素的具体区别是什么？行内元素的 padding 和 margin 可设置吗?

块级元素(block)特性：

总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;
宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;
内联元素(inline)特性：
和相邻的内联元素在同一行;
宽度(width)、高度(height)、内边距的 top/bottom(padding-top/padding-bottom)和外边距的 top/bottom(margin-top/margin-bottom)都不可改变（也就是 padding 和 margin 的 left 和 right 是可以设置的），就是里面文字或图片的大小。

那么问题来了，浏览器还有默认的天生 inline-block 元素（拥有内在尺寸，可设置高宽，但不会自动换行），有哪些?

答案：<input>、<img>、<button>、<textarea>、<label>

56. 什么是外边距重叠？重叠的结果是什么?

答案：

外边距重叠就是 margin-collapse。

在 CSS 当中，相邻的两个盒子（可能是兄弟关系也可能是祖先关系）的外边距可以结合成一个单独的外边距。这种合并外边距的方式被称为折叠，并且因而所结合成的外边距称为折叠外边距。

折叠结果遵循下列计算规则：

1. 两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。
2. 两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。
3. 两个外边距一正一负时，折叠结果是两者的相加的和。

58、描述一个"reset"的 CSS 文件并如何使用它。知道 normalize.css 吗？你了解他们的不同之处？

重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置 CSS 文件并知道如何使用它们。他们是盲目的在做还是知道为什么这么做呢？原因是不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，或者更有戏剧性的性发生。

你可能会用 Normalize 来代替你的重置样式文件。它没有重置所有的样式风格，但仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，但又不扰乱其他的东西（如粗体的标题）。

在这一方面，无法做每一个复位重置。它也确实有些超过一个重置，它处理了你永远都不用考虑的怪癖，像 HTML 的 audio 元素不一致或 line-height 不一致。

57. 说 display 属性有哪些？可以做什么？

display:block 行内元素转换为块级元素
display:inline 块级元素转换为行内元素
display:inline-block 转为内联元素

58. 哪些 css 属性可以继承？

可继承： font-size font-family color, ul li dl dd dt;
不可继承： border padding margin width height ;

59. css 优先级算法如何计算？

!important > id > class > 标签
!important 比 内联优先级高

*优先级就近原则，样式定义最近者为准;
*以最后载入的样式为准;

60. b 标签和 strong 标签,i 标签和 em 标签的区别?

后者有语义，前者则无。

61. 有那些行内元素、有哪些块级元素、盒模型?

1.内联元素(inline element)

a – 锚点

abbr – 缩写

acronym – 首字

b – 粗体(不推荐)

big – 大字体

br – 换行

em – 强调

font – 字体设定(不推荐)

i – 斜体

img – 图片

input – 输入框

label – 表格标签

s – 中划线(不推荐)

select – 项目选择

small – 小字体文本

span – 常用内联容器，定义文本内区块

strike – 中划线

strong – 粗体强调

sub – 下标

sup – 上标

textarea – 多行文本输入框

tt – 电传文本

u – 下划线

var – 定义变量

2、块级元素

address – 地址

blockquote – 块引用

center – 举中对齐块

dir – 目录列表

div – 常用块级容易，也是 css layout 的主要标签

dl – 定义列表

fieldset – form 控制组

form – 交互表单

h1 – 大标题

h2 – 副标题

h3 – 3 级标题

h4 – 4 级标题

h5 – 5 级标题

h6 – 6 级标题

hr – 水平分隔线

isindex – input prompt

menu – 菜单列表

noframes – frames 可选内容，（对于不支持 frame 的浏览器显示此区块内容）

noscript – ）可选脚本内容（对于不支持 script 的浏览器显示此内容）

ol – 排序表单

p – 段落

pre – 格式化文本

table – 表格

ul – 非排序列表

3.CSS 盒子模型包含四个部分组成：

内容、填充（padding）、边框（border）、外边界（margin）。

62. 有哪些选择符，优先级的计算公式是什么？行内样式和！important 哪个优先级高?

#ID > .class > 标签选择符 !important 优先级高

63. 我想让行内元素跟上面的元素距离 10px，加 margin-top 和 padding-top 可以吗?

margin-top,padding-top 无效

64. CSS 的盒模型由什么组成?

内容，border ,margin，padding

65. 说说 display 属性有哪些？可以做什么?

display:block 行内元素转换为块级元素

display:inline 块级元素转换为行内元素

display:inline-block 转为内联元素

66. 哪些 css 属性可以继承?

可继承： font-size font-family color, ul li dl dd dt;

不可继承： border padding margin width height ;

67. css 优先级算法如何计算?

!important > id > class > 标签

!important 比 内联优先级高

* 优先级就近原则，样式定义最近者为准;

* 以最后载入的样式为准;

二、JS 基础

1. javascript 的 typeof 返回哪些数据类型

```
alert(typeof [1, 2]); //object
alert(typeof 'leipeng'); //string
var i = true;
alert(typeof i); //boolean
alert(typeof 1); //number
var a;
alert(typeof a); //undefined
function a(){;};
alert(typeof a) //function
```

2. 例举 3 种强制类型转换和 2 种隐式类型转换?

强制（parseInt(),parseFloat(),Number()）

隐式（== ,!!）

3. split() 、join() 的区别

前者是切割成数组的形式，后者是将数组转换成字符串

4. 数组方法 pop() push() unshift() shift()

Push()尾部添加 pop()尾部删除

Unshift()头部添加 shift()头部删除

5. 事件绑定和普通事件有什么区别

普通添加事件的方法：

```
var btn = document.getElementById("hello");
btn.onclick = function(){
    alert(1);
}
btn.onclick = function(){
    alert(2);
}
```

执行上面的代码只会 alert 2

事件绑定方式添加事件：

```
var btn = document.getElementById("hello");
btn.addEventListener("click",function(){
    alert(1);
},false);
btn.addEventListener("click",function(){
```

```
    alert(2);
},false);
```

执行上面的代码会先 alert 1 再 alert 2

普通添加事件的方法不支持添加多个事件，最下面的事件会覆盖上面的，而事件绑定（addEventListener）方式添加事件可以添加多个。

addEventListener 不兼容低版本 IE

普通事件无法取消

addEventLisntener 还支持事件冒泡+事件捕获

6. IE 和 DOM 事件流的区别

1.执行顺序不一样、

2.参数不一样

3.事件加不加 on

4.this 指向问题

7. IE 和标准下有哪些兼容性的写法

Var ev = ev || window.event

document.documentElement.clientWidth || document.body.clientWidth

Var target = ev.srcElement||ev.target

8. call 和 apply 的区别

call 方法:

语法：call(thisObj，Object1,Object2...)

定义：调用一个对象的一个方法，以另一个对象替换当前对象。

说明:

call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

apply 方法：

语法：apply(thisObj，[argArray])

定义：应用某一对象的一个方法，用另一个对象替换当前对象。

说明：

如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。

如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

9. b 继承 a 的方法

```
function A( age, name ){
  this.age = age;
  this.name = name;
}

A.prototype.show = function(){
  alert('父级方法');
}

function B(age,name,job){
  A.apply( this, arguments );
  this.job = job;
}
```

```
B.prototype = new A();
var b = new A(14,'侠客行');
var a = new B(15,'狼侠','侠客');
```

10. 如何阻止事件冒泡和默认事件

canceBubble()只支持 IE,return false,stopPropagation()

11. 添加 删除 替换 插入到某个接点的方法

obj.appendChid()

obj.insertBefore()

obj.replaceChild()

obj.removeChild()

12. javascript 的本地对象，内置对象和宿主对象

本地对象为 array obj regexp 等可以 new 实例化

内置对象为 gload Math 等不可以实例化的

宿主为浏览器自带的 document,window 等

13. window.onload 和 document ready 的区别

window.onload 是在 dom 文档树加载完和所有文件加载完之后执行一个函数

Document.ready 原生种没有这个方法，jquery 中有 $().ready(function),在 dom 文档树加载完之后执行一个函数（注意，这里面的文档树加载完不代表全部文件加载完）。

$(document).ready 要比 window.onload 先执行

window.onload 只能出来一次，$(document).ready 可以出现多次

14.”==”和“===”的不同

前者会自动转换类型

后者不会

15. javascript 的同源策略

一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、议和端口号的组合

16. JavaScript 是一门什么样的语言，它有哪些特点?

没有标准答案。

javaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。JavaScript 兼容于 ECMA 标准，因此也称为 ECMAScript。

基本特点

1.是一种解释性脚本语言（代码不进行预编译）。

2.主要用来向 HTML（标准通用标记语言下的一个应用）页面添加交互行为。

3.可以直接嵌入 HTML 页面，但写成单独的 js 文件有利于结构和行为的分离。

4.跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如 Windows、Linux、Mac、Android、iOS 等）。

17. JavaScript 的数据类型都有什么?

基本数据类型：String,boolean,Number,Undefined, Null

引用数据类型：Object(Array,Date,RegExp,Function)

那么问题来了，如何判断某变量是否为数组数据类型?

方法一.判断其是否具有“数组性质”，如 slice()方法。可自己给该变量定义 slice 方法，故有时会失效

方法二.obj instanceof Array 在某些 IE 版本中不正确

方法三.方法一二皆有漏洞，在 ECMA Script5 中定义了新方法 Array.isArray(), 保证其兼容性，最好的方法如下：

```
if(typeof Array.isArray==="undefined")
{
  Array.isArray = function(arg){
      return Object.prototype.toString.call(arg)==="[object Array]"
    };
}
```

18. 已知 ID 的 Input 输入框，希望获取这个输入框的输入值，怎么做？(不使用第三方框架)

```
document.getElementById(“ID”).value
```

19.

19. 希望获取到页面中所有的 checkbox 怎么做？(不使用第三方框架)

```
var domList = document.getElementsByTagName( 'input' )
var checkBoxList = [];
var len = domList.length;      //缓存到局部变量
while (len--) {      //使用 while 的效率会比 for 循环更高
    if (domList[len].type ==  'checkbox' ) {
        checkBoxList.push(domList[len]);
    }
}
```

20.

20. 设置一个已知 ID 的 DIV 的 html 内容为 xxxx，字体颜色设置为黑色(不使用第三方框架)

```
var dom = document.getElementById( "ID" );
dom.innerHTML =  "xxxx"
dom.style.color =  "#000"
```

21.

21. 当一个 DOM 节点被点击时候，我们希望能够执行一个函数，应该怎么做?

直接在 DOM 里绑定事件：

在 JS 里通过 onclick 绑定：xxx.onclick = test

通过事件添加进行绑定：addEventListener(xxx, ‘click’, test)

那么问题来了，Javascript 的事件流模型都有什么?

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM 事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

22. 看下列代码输出为何？解释原因。

```
var a;
alert(typeof a); // undefined
alert(b); // 报错
```

23.

解释：Undefined 是一个只有一个值的数据类型，这个值就是“undefined”，在使用 var 声明变量但并未对其赋值进行初始化时，这个变量的值就是 undefined。而 b 由于未声明将报错。注意未申明的变量和声明了未赋值的是不一样的。

23. 看下列代码,输出什么？解释原因。

```
var a = null;
alert(typeof a); //object
```

24.

解释：null 是一个只有一个值的数据类型，这个值就是 null。表示一个空指针对象，所以用 typeof 检测会返回”object”。

24. 看下列代码,输出什么？解释原因。

```
var undefined;
undefined == null; // true
1 == true;  // true
2 == true;  // false
0 == false;  // true
0 == '';    // true
NaN == NaN;  // false
[] == false; // true
[] == ![];  // true
```

25.

- undefined 与 null 相等，但不恒等（===）

一个是 number 一个是 string 时，会尝试将 string 转换为 number

尝试将 boolean 转换为 number，0 或 1

尝试将 Object 转换成 number 或 string，取决于另外一个对比量的类型

所以，对于 0、空字符串的判断，建议使用 “===” 。“===”会先判断两边的值类型，类型不匹配时为 false。

那么问题来了，看下面的代码，输出什么，foo 的值为什么?

```
var foo = "11"+2-"1";
console.log(foo);
console.log(typeof foo);
```

执行完后 foo 的值为 111，foo 的类型为 String。

25. 看代码给答案。

```
var a = new Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
```

26.

答案：2（考察引用数据类型细节）

26. 已知数组 var stringArray = [“This”，“is”，“Baidu”，“Campus”]，Alert 出”This is Baidu Campus”。

答案：alert(stringArray.join(“”))

27. 已知有字符串 foo=”get-element-by-id”,写一个 function 将其转化成驼峰表示法”getElementById”。

```
function combo(msg){
    var arr=msg.split("-");
    for(var i=1;i<arr.length;i++){
        arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);
    }
    msg=arr.join("");
    return msg;
}
```

28.

(考察基础 API)

28. var numberArray = [3,6,2,4,1,5]; (考察基础 API)

1) 实现对该数组的倒排，输出[5,1,4,2,6,3]

numberArray.reverse()

2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

numberArray.sort(function(a,b){return b-a})

29. 输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2014 年 9 月 26 日，则输出 2014-09-26

```
var d = new Date();
// 获取年，getFullYear()返回 4 位的数字
var year = d.getFullYear();
// 获取月，月份比较特殊，0 是 1 月，11 是 12 月
var month = d.getMonth() + 1;
// 变成两位
month = month < 10 ? '0' + month : month;
// 获取日
var day = d.getDate();
day = day < 10 ? '0' + day : day;
alert(year + '-' + month + '-' + day);
```

30.

30. 将字符串“<tr><td>{$id}</td><td>{$name}</td></tr>”中的 {$id}替换成 10，{$name}替换成 Tony （使用正则表达式）

答案：

```
"<tr><td>{$id}</td><td>{$id}_{$name}</td></tr>".replace(/{\$id}/g, '10').replac
e(/{\$name}/g, 'Tony');
```

31. 为了保证页面输出安全，我们经常需要对一些特殊的字符进行转义，请写一个函数 escapeHtml，将<, >, &, “进行转义

```
function escapeHtml(str) {
return str.replace(/[<>”&]/g, function(match) {
  switch (match) {
        case “<”:
          return “&lt;”;
        case “>”:
          return “&gt;”;
        case “&”:
          return “&amp;”;
        case “\””:
          return “&quot;”;
    }
 });
}
```

32.

32. foo = foo||bar，这行代码是什么意思？为什么要这样写？

答案：if(!foo) foo = bar; //如果 foo 存在，值不变，否则把 bar 的值赋给 foo。

短路表达式：作为”&&”和”||”操作符的操作数表达式，这些表达式在进行求值时，只要最终的结果已经可以确定是真或假，求值过程便告终止，这称之为短路求值。

33. 看下列代码，将会输出什么?(变量声明提升)

```
var foo = 1;
(function(){
    console.log(foo);
    var foo = 2;
    console.log(foo);
})()
```

34.

答案：输出 undefined 和 2。上面代码相当于：

```
var foo = 1;
(function(){
    var foo;
    console.log(foo); //undefined
    foo = 2;
    console.log(foo); // 2;
})()
```

函数声明与变量声明会被 JavaScript 引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分。

34. 用 js 实现随机选取 10–100 之间的 10 个数字，存入一个数组，并排序。

```
function randomNub(aArray, len, min, max) {
        if (len >= (max - min)) {
            return '超过' + min + '-' + max + '之间的个数范围' + (max - min - 1) + '个的总数';
        }
        if (aArray.length >= len) {
            aArray.sort(function(a, b) {
                return a - b
            });
            return aArray;
        }
        var nowNub = parseInt(Math.random() * (max - min - 1)) + (min + 1);
        for (var j = 0; j < aArray.length; j++) {
            if (nowNub == aArray[j]) {
                randomNub(aArray, len, min, max);
                return;
            }
        }
        aArray.push(nowNub);
        randomNub(aArray, len, min, max);
        return aArray;
    }
```

```
var arr=[];
randomNub(arr,10,10,100);
```

35.

35. 把两个数组合并，并删除第二个元素。

```
var array1 = ['a','b','c'];
var bArray = ['d','e','f'];
var cArray = array1.concat(bArray);
cArray.splice(1,1);
```

36.

36. 怎样添加、移除、移动、复制、创建和查找节点（原生 JS，实在基础，没细写每一步）

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()　　//添加

removeChild()　　//移除

replaceChild()　　//替换

insertBefore()　　//插入

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

37.有这样一个URL：http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e，请写一段JS程序提取URL中的各个GET参数(参数名和参数个数不确定)，将其按key-value形式返回到一个json结构中，如{a:‘1’, b:‘2’, c:‘’, d:‘xxx’, e:undefined}。

答案：

```
function serilizeUrl(url) {
    var urlObject = {};
    if (/\?/.test(url)) {
        var urlString = url.substring(url.indexOf("?") + 1);
        var urlArray = urlString.split("&");
        for (var i = 0, len = urlArray.length; i < len; i++) {
            var urlItem = urlArray[i];
            var item = urlItem.split("=");
            urlObject[item[0]] = item[1];
        }
        return urlObject;
    }
    return null;
}
```

38. 正则表达式构造函数 var reg=new RegExp(“xxx”)与正则表达字面量 var reg=//有什么不同？匹配邮箱的正则表达式?

答案：当使用 RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。

邮箱的正则匹配：

```
var regMail = /^([a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+((.[a-zA-Z0-9_-]{2,3}){1,2})$/;
```

39. 看下面代码，给出输出结果。

```
for(var i=1;i<=3;i++){
  setTimeout(function(){
      console.log(i);
  },0);
};
```

40.

答案：4 4 4。

原因：Javascript 事件处理器在线程空闲之前不会运行。追问，如何让上述代码输出 1 2 3?

```
for(var i=1;i<=3;i++){
  setTimeout((function(a){  //改成立即执行函数
      console.log(a);
  })(i),0);
};

1          //输出
2
3
```

40. 写一个 function，清除字符串前后的空格。（兼容所有浏览器）

使用自带接口 trim()，考虑兼容性：

```
if (!String.prototype.trim) {
 String.prototype.trim = function() {
 return this.replace(/^\s+/, "").replace(/\s+$/,"");
 }
}

// test the function
var str = " \t\n test string ".trim();
alert(str == "test string"); // alerts "true"
```

41. Javascript 中 callee 和 caller 的作用?

caller 是返回一个对函数的引用，该函数调用了当前函数；

callee 是返回正在被执行的 function 函数，也就是所指定的 function 对象的正文。

那么问题来了？如果一对兔子每月生一对兔子；一对新生兔，从第二个月起就开始生兔子；假定每对兔子都是一雌一雄，试问一对兔子，第 n 个月能繁殖成多少对兔子？（使用 callee 完成）

```
var result=[];
function fn(n){  //典型的斐波那契数列
  if(n==1){
      return 1;
  }else if(n==2){
        return 1;
  }else{
      if(result[n]){
            return result[n];
      }else{
            //argument.callee()表示 fn()
            result[n]=arguments.callee(n-1)+arguments.callee(n-2);
            return result[n];
      }
  }
}
```

42. Javascript 中, 以下哪条语句一定会产生运行错误? 答案(B C)

var _变量=NaN; B、var 0bj = []; C、var obj = //; D、var obj = {};

43. 以下两个变量 a 和 b，a+b 的哪个结果是 NaN? 答案(AC)

A、var a=undefined; b=NaN

B、var a= '123'; b=NaN

C、var a =undefined , b =NaN

var a=NaN , b='undefined'

44. var a=10; b=20; c=4; ++b+c+a++ 以下哪个结果是正确的? 答案(B)

A. 34 B、35 C、36 D、37

45. 下面的 JavaScript 语句中，(D) 实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

A. for(vari=0;i< form1.elements.length;i++) {

if(form1.elements.type==" text")

form1.elements.value=";}

B. for(vari=0;i<document.forms.length;i++) {

if(forms[0].elements.type==" text")

forms[0].elements.value=";

}

C. if(document.form.elements.type==" text")

form.elements.value=” ";

D. for(vari=0;i<document.forms.length; i++){

for(var j=0;j<document.forms.elements.length; j++){

if(document.forms.elements[j].type==” text”)

document.forms.elements[j].value=” ";

}

}

46. 要将页面的状态栏中显示“已经选中该文本框”，下列 JavaScript 语句正确的是（A）

A. window.status=”已经选中该文本框”

B. document.status=”已经选中该文本框”

C. window.screen=”已经选中该文本框”

D. document.screen=”已经选中该文本框”

47. 以下哪条语句会产生运行错误：（AD）

A.var obj = ();B.var obj = [];C.var obj = {};D.var obj = //;

48. 以下哪个单词不属于 javascript 保留字：（B）

A.withB.parentC.classD.void

49. 请选择结果为真的表达式：（C）

A.null instanceof ObjectB.null === undefinedC.null == undefinedD.NaN == NaN

50. Javascript 中，如果已知 HTML 页面中的某标签对象的 id=”username” ，用 ____document.getElementById（‘username’）___ _方法获得该标签对象。

51. typeof 运算符返回值中有一个跟 javascript 数据类型不一致，它是 _____”function”_____。

52. 定义了一个变量，但没有为该变量赋值，如果 alert 该变量，javascript 弹出的对话框中显示___undefined_____ 。

53. 分析代码，得出正确的结果。

```
var a=10, b=20 , c=30;
    ++a;
    a++;
    e=++a+(++b)+(c++)+a++;
    alert(e);
```

弹出提示对话框：77

54. 写出函数 DateDemo 的返回结果，系统时间假定为今天

```
function DateDemo(){
 var d, s="今天日期是：";
 d = new Date();
s += d.getMonth() +1+ "/";
```

```
s += d.getDate() + "/";
s += d.getFullYear();
return s;}
```

结果：今天日期是：7/17/2010

55. 写出程序运行的结果?

```
for(i=0, j=0; i<10, j<6; i++, j++){
k = i + j;}
```

结果：10

56. 阅读以下代码，请分析出结果：

```
var arr = new Array(1 ,3 ,5);
arr[4]='z';
arr2 = arr.reverse();
arr3 = arr.concat(arr2);
alert(arr3);
```

弹出提示对话框：z,,5,3,1,z,,5,3,1

57. 补充按钮事件的函数，确认用户是否退出当前页面，确认之后关闭窗?

```
<html>
<head>
<script type=”text/javascript” >
function closeWin(){
//在此处添加代码
if(confirm(“确定要退出吗？”)){
```

```
window.close();
}
}
</script>
</head>
<body>
<input type=”button”value=”关闭窗口”onclick=”closeWin()”/>
</body>
</html>
```

58. 写出简单描述 html 标签（不带属性的开始标签和结束标签）的正则表达式，并将以下字符串中的 html 标签去除掉

```
var str = “<div>这里是 div<p>里面的段落</p></div>”;
//
<scripttype=”text/javascript”>
var reg = /<\/?\w+\/?>/gi;
var str = “<div>这里是 div<p>里面的段落</p></div>”;
alert(str.replace(reg,”"));
</script>
```

59. 完成 foo()函数的内容，要求能够弹出对话框提示当前选中的是第几个单选框。

```
<html>
<head>
```

```
<metahttp-equiv=”Content-Type” content=”text/html;charset=utf-8″/>
</head>
<body>
<script type=”text/javascript” >
function foo() {
//在此处添加代码
var rdo =document.form1.radioGroup;
for(var i =0 ;i<rdo.length;i++){
if(rdo.checked){
alert(“您选择的是第”+(i+1)+”个单选框”);
}
}
}
</script>
<body>
<form name=”form1″>
<input type=”radio” name=”radioGroup”/>
<input type=”radio” name=”radioGroup”/>
<input type=”radio” name=”radioGroup”/>
<input type=”radio” name=”radioGroup”/>
<input type=”submit”/>
</form>
</body>
</html>
```

60. 完成函数 showImg()，要求能够动态根据下拉列表的选项变化，更新图片的显示

```
<body>
<script type=”text/javascript” >
function showImg (oSel) {
//在此处添加代码
var str = oSel.value;
document.getElementById(“pic”).src= str+”.jpg”;
}
</script>
<img id=” pic” src=” img1.jpg” width=” 200” height=” 200” />
<br />
<select id=” sel” >
<option value=” img1 “>城市生活</option>
<option value=” img2 “>都市早报</option>
<option value=” img3 “>青山绿水</option>
</select></body>
```

61. 截取字符串 abcdefg 的 efg

```
alert('abcdefg'.substring(4));
```

62. 列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个

对象：window, document, location, screen, history, navigator

方法：alert(), confirm(), prompt(), open(), close()

63. 简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法并做简单说明

Document.getElementById 根据元素 id 查找元素

Document.getElementByName 根据元素 name 查找元素

Document.getElementTagName 根据指定的元素名查找元素

64. 希望获取到页面中所有的 checkbox 怎么做？(不使用第三方框架)

```
var domList = document.getElementsByTagName('input')
var checkBoxList = [];
var len = domList.length;      //缓存到局部变量
while (len--) {      //使用 while 的效率会比 for 循环更高
    if (domList[len].type ==  'checkbox' ) {
        checkBoxList.push(domList[len]);
    }
}
```

65. 简述创建函数的几种方式

第一种（函数声明）：

```
function sum1(num1,num2){
  return num1+num2;
}
```

第二种（函数表达式）：

```
var sum2 = function(num1,num2){
  return num1+num2;
}
```

第三种（函数对象方式）：

```
var sum3 = new Function("num1","num2","return num1+num2");
```

66. Javascript 如何实现继承?

1. 构造继承法

2. 原型继承法

3.实例继承法

67. Javascript 创建对象的几种方式?

1、var obj = {};（使用 json 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

};

2、var obj = new Object();（使用 Object 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

};

3、通过函数创建对象。

(1)、使用 this 关键字

如：var obj = function (){

this.name ='张三';

this.age = 19;

this.action = function ()

{

alert('吃饭');

};

}

(2)、使用 prototype 关键字

如：function obj (){}

obj.prototype.name ='张三';

obj.prototype.action=function ()

{

alert('吃饭');

};

4、通过 Window 创建对象。

如：window.name = ''张三';

window.age = 19;

window.action= function()

{

alert('吃饭');

};

5、使用内置对象创建对象。

如：var str = new String("实例初始化 String");

var str1 = "直接赋值的 String";

var func = new Function("x","alert(x)");//示例初始化 func

var obj = new Object();//示例初始化一个 Object

68. iframe 的优缺点?

优点:

1. 解决加载缓慢的第三方内容如图标和广告等的加载问题

2. Security sandbox

3. 并行加载脚本

缺点:

1. iframe 会阻塞主页面的 Onload 事件

2. 即时内容为空，加载也需要时间

3. 没有语意

69. 请你谈谈 Cookie 的弊端?

缺点:

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

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

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

70. js 延迟加载的方式有哪些?

1. defer 和 async

2. 动态创建 DOM 方式（创建 script，插入到 DOM 中，加载完毕后 callBack）

3. 按需异步载入 js

71. 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

72. 哪些操作会造成内存泄漏?

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

1. 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。

2. 闭包

3. 控制台日志

4. 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）

73. 判断一个字符串中出现次数最多的字符，统计这个次数

```
答：var str = 'asdfssaaasasasasaa';
var json = {};
for (var i = 0; i < str.length; i++) {
    if(!json[str.charAt(i)]){
        json[str.charAt(i)] = 1;
    }else{
        json[str.charAt(i)]++;
```

```
                                                                    }
};
var                 iMax                =                   0;
var                 iIndex              =                   '';
for(var             i                   in                  json){
                                                    if(json[i]>iMax){
                                            iMax    =   json[i];
                                                iIndex  =   i;
                                                                    }
}
alert('出现次数最多的是:'+iIndex+'出现'+iMax+'次');
```

74. 写一个获取非行间样式的函数

```
function getStyle(obj,attr,value)
{
    if(!value)
    {
        if(obj.currentStyle)
        {
            return obj.currentStyle(attr);
        }
        else{
            obj.getComputedStyle(attr,false);
        }
    }
```

```
    else
    {
        obj.style[attr] = value;
    }
}
```

75. 事件委托是什么

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

76. 闭包是什么，有什么特性，对页面有什么影响

答：我的理解是，闭包就是能够读取其他函数内部变量的函数。在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

```
function outer(){
    var num = 1;
    function inner(){
        var n = 2;
        alert(n + num);
    }
    return inner;
}
outer()();
```

http://blog.csdn.net/gaoshanwudi/article/details/7355794 此链接可查看（问这个问题的不是一个公司）

77. 解释 jsonp 的原理，以及为什么不是真正的 ajax

动态创建 script 标签，回调函数

Ajax 是页面无刷新请求数据操作

78. javascript 的本地对象，内置对象和宿主对象

本地对象为 array obj regexp 等可以 new 实例化

内置对象为 gload Math 等不可以实例化的

宿主为浏览器自带的 document,window 等

79. 字符串反转，如将 '12345678' 变成 '87654321'

```
//大牛做法；
//思路：先将字符串转换为数组 split()，利用数组的反序函数 reverse()颠倒数组，再利用 jion() 转换为
串
var str = '12345678';
str = str.split('').reverse().join('');
```

80.

80. 将数字 12345678 转化成 RMB 形式 如： 12,345,678

```
//个人方法;
//思路：先将数字转为字符， str= str + '' ;
//利用反转函数，每三位字符加一个 ','最后一位不加； re()是自定义的反转函数，最后再反转回去!
function re(str) {
    str += '';
    return str.split("").reverse().join("");
}

function toRMB(num) {
      var tmp='';
    for (var i = 1; i <= re(num).length; i++) {
        tmp += re(num)[i - 1];
        if (i % 3 == 0 && i != re(num).length) {
            tmp += ',';
        }
    }
    return re(tmp);
}
```

81.

81. 生成 5 个不同的随机数；

```
//思路：5 个不同的数，每生成一次就和前面的所有数字相比较，如果有相同的，则放弃当前生成的数
var num1 = [];
for(var i = 0; i < 5; i++){
    num1[i] = Math.floor(Math.random()*10) + 1; //范围是 [1, 10]
    for(var j = 0; j < i; j++){
        if(num1[i] == num1[j]){
            i--;
        }
    }
}
```

82.

82. 去掉数组中重复的数字 方法一；

```
//思路：每遍历一次就和之前的所有做比较，不相等则放入新的数组中！
//这里用的原型 个人做法；
Array.prototype.unique = function(){
    var len = this.length,
        newArr = [],
        flag = 1;
    for(var i = 0; i < len; i++, flag = 1){
        for(var j = 0; j < i; j++){
            if(this[i] == this[j]){
                flag = 0;       //找到相同的数字后，不执行添加数据
            }
        }
        flag ? newArr.push(this[i]) : '';
    }
    return newArr;
}
```

83.

方法二：

var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];

```
Array.prototype.unique2 = function()
{
        var n = []; //一个新的临时数组
        for(var i = 0; i < this.length; i++) //遍历当前数组
        {
                //如果当前数组的第 i 已经保存进了临时数组，那么跳过，
                //否则把当前项 push 到临时数组里面
                if (n.indexOf(this[i]) == -1) n.push(this[i]);
        }
        return n;
}
```

var newArr2=arr.unique2(arr);

alert(newArr2); //输出 1,2,3,4,5,6,9,25

83. 阶乘函数；

```
//原型方法
Number.prototype.N = function(){
    var re = 1;
    for(var i = 1; i <= this; i++){
        re *= i;
    }
    return re;
}
var num = 5;
```

```
alert(num.N());
```

84.

84. window.location.search() 返回的是什么?

答：查询(参数)部分。除了给动态语言赋值以外，我们同样可以给静态页面,并使用 javascript 来获得相信应的参数值

返回值：?ver=1.0&id=timlq 也就是问号后面的!

85. window.location.hash 返回的是什么?

答：锚点， 返回值：#love；

86. window.location.reload() 作用?

答：刷新当前页面。

87.、javascript 中的垃圾回收机制?

答：在 Javascript 中，如果一个对象不再被引用，那么这个对象就会被 GC 回收。如果两个对象互相引用，而不再 被第 3 者所引用，那么这两个互相引用的对象也会被回收。因为函数 a 被 b 引用，b 又被 a 外的 c 引用，这就是为什么 函数 a 执行后不会被回收的原因。

88. 看题做答：

```
function f1(){
    var tmp = 1;
    this.x = 3;
    console.log(tmp);    //A
    console.log(this.x);       //B
}
var obj = new f1(); //1
console.log(obj.x)     //2
console.log(f1());       //3
```

89.

分析：

这道题让我重新认识了对象和函数，首先看代码（1），这里实例话化了 f1 这个类。相当于执行了 f1 函数。所以这个时候 A 会输出 1， 而 B 这个时候的 this 代表的是 实例化的当前对象 obj B 输出 3.。 代码（2）毋庸置疑会输出 3， 重点 代码（3）首先这里将不再是一个类，它只是一个函数。那么 A 输出 1， B 呢？这里的 this 代表的其实就是 window 对象，那么 this.x 就是一个全局变量 相当于在外部 的一个全局变量。所以 B 输出 3。最后代码由于 f 没有返回值那么一个函数如果没返回值的话，将会返回 underfined ，所以答案就是： 1， 3， 3， 1， 3， underfined 。

89. 下面输出多少?

```
var o1 = new Object();
var o2 = o1;
o2.name = "CSSer";
console.log(o1.name);
```

90.

如果不看答案，你回答真确了的话，那么说明你对 javascript 的数据类型了解的还是比较清楚了。js 中有两种数据类型，分别是：基本数据类型和引用数据类型（object Array）。对于保存基本类型值的变量，变量是按值访问的，因为我们操作的是变量实际保存的值。对于保存引用类型值的变量，变量是按引用访问的，我们操作的是变量值所引用（指向）的对象。答案就清楚了： //CSSer;

90. 再来一个

```
function changeObjectProperty (o) {
    o.siteUrl = "http://www.csser.com/";
    o = new Object();
    o.siteUrl = "http://www.popcg.com/";
}
var CSSer = new Object();
changeObjectProperty(CSSer);
console.log(CSSer.siteUrl); //
```

91.

如果 CSSer 参数是按引用传递的，那么结果应该是"http://www.popcg.com/"，但实际结果却仍是"http://www.csser.com/"。事实是这样的：在函数内部修改了引用类型值的参数，该参数值的原始引用保持不变。我们可以把参数想象成局部变量，当参数被重写时，这个变量引用的就是一个局部变量，局部变量的生存期仅限于函数执行的过程中，函数执行完毕，局部变量即被销毁以释放内存。

（补充：内部环境可以通过作用域链访问所有的外部环境中的变量对象，但外部环境无法访问内部环境。每个环境都可以向上搜索作用域链，以查询变量和函数名，反之向下则不能。）

91. a 输出多少?

```
var a = 6;
setTimeout(function () {
    var a = 666;
    alert(a);      // 输出 666，
}, 1000);
```

92.

因为 var a = 666;定义了局部变量 a，并且赋值为 666，根据变量作用域链，全局变量处在作用域末端，优先访问了局部变量，从而覆盖了全局变量 。

```
var a = 6;
setTimeout(function () {
    alert(a);      // 输出 undefined
    var a = 666;
}, 1000);
```

因为 var a = 666;定义了局部变量 a，同样覆盖了全局变量，但是在 alert(a);之前 a 并未赋值，所以输出 undefined。

```
var a = 6;
setTimeout(function(){
    alert(a);
    var a = 66;
}, 1000);
a = 666;
```

```
alert(a);
// 666, undefined;
```

记住： 异步处理，一切 OK 声明提前

92. 看程序，写结果

```
function setN(obj){
    obj.name='屌丝';
    obj = new Object();
    obj.name = '腐女';
};
var per = new Object();
setN(per);
alert(per.name);  //屌丝 内部
```

93.

93. JS 的继承性

```
window.color = 'red';
var o = {color: 'blue'};
function sayColor(){
    alert(this.color);
}
sayColor(); //red
sayColor.call(this); //red this-window 对象
sayColor.call(window); //red
sayColor.call(o); //blue
```

94.

94. 精度问题: JS 精度不能精确到 0.1 所以 。。。。同时存在于值和差值中

```
var n = 0.3,m = 0.2, i = 0.2, j = 0.1;
alert((n - m) == (i - j)); //false
alert((n-m) == 0.1); //false
alert((i-j)==0.1); //true
```

95.

95. 加减运算

```
alert('5'+3); //53 string
alert('5'+'3'); //53 string
alert('5'-3); //2 number
alert('5'-'3'); //2 number
```

96.

96. 什么是同源策略?

指： 同协议、端口、域名的安全策略，由王景公司提出来的安全协议!

97. 为什么不能定义 1px 左右的 div 容器?

IE6 下这个问题是因为默认的行高造成的，解决的方法也有很多，例如：
overflow:hidden | zoom:0.08 | line-height:1px

98. 结果是什么?

```
function foo(){
    foo.a = function(){alert(1)};
    this.a = function(){alert(2)};
    a = function(){alert(3)};
    var a = function(){alert(4)};
};
foo.prototype.a = function(){alert(5)};
foo.a = function(){alert(6)};
```

```
foo.a(); //6
var obj = new foo();
obj.a(); //2
foo.a(); //1
```

99.

99. 输出结果

```
var a = 5;
function test(){
    a = 0;
    alert(a);
    alert(this.a); //没有定义 a 这个属性
    var a;
    alert(a)
}
test(); // 0, 5, 0
new test(); // 0, undefined, 0 //由于类它自身没有属性 a， 所以是 undefined
```

100.

100. 计算字符串字节数：

```
new function(s){
    if(!arguments.length||!s) return null;
    if(""==s) return 0;
    var l=0;
    for(var i=0;i<s.length;i++){
        if(s.charCodeAt(i)>255) l+=2; else l+=1;  //charCodeAt()得到的是 unCode 码
    }    //汉字的 unCode 码大于 255bit 就是两个字节
    alert(l);
}("hello world!");
```

101.

101. 结果是：

```
var bool = !!2; alert(bool)；//true;
```

102.

双向非操作可以把字符串和数字转换为布尔值。

102. 声明对象，添加属性，输出属性

```
var obj = {
    name: 'leipeng',
    showName: function(){
        alert(this.name);
    }
}
obj.showName();
```

103.

103. 匹配输入的字符：第一个必须是字母或下划线开头，长度 5-20

```
var reg = /^[a-zA-Z_][a-zA-Z0-9_]{5,20}/,
    name1 = 'leipeng',
    name2 = '0leipeng',
    name3 = '你好 leipeng',
    name4 = 'hi';

alert(reg.test(name1));
alert(reg.test(name2));
alert(reg.test(name3));
alert(reg.test(name4));
```

104.

104. 检测变量类型

```
function checkStr(str){
    return str =='string';
}


console.log(checkStr("aaa"));
```

105. 如何在 HTML 中添加事件，几种方法?

1、标签之中直接添加 onclick="fun()";

2、JS 添加 Eobj.onclick = method;

3、现代事件 IE： obj.attachEvent('onclick', method)；

FF: obj.addEventListener('click', method, false);

106. BOM 对象有哪些，列举 window 对象?

1、window 对象 ，是 JS 的最顶层对象，其他的 BOM 对象都是 window 对象的属性；

2、document 对象，文档对象；

3、location 对象，浏览器当前 URL 信息；

4、navigator 对象，浏览器本身信息；

5、screen 对象，客户端屏幕信息；

6、history 对象，浏览器访问历史信息；

107. 请问代码实现 outerHTML

//说明：outerHTML 其实就是 innerHTML 再加上本身；

```
    Object.prototype.outerHTML = function(){
        var innerCon = this.innerHTML, //获得里面的内容
            outerCon = this.appendChild(innerCon); //添加到里面
        alert(outerCon);
      }
```

演示代码：

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Document</title>
 </head>
 <body>
  <div id="outer">
    hello
  </div>
 <script>
  Object.prototype.outerHTML = function(){
  var innerCon = this.innerHTML, //获得里面的内容
  outerCon = this.appendChild(innerCon); //添加到里面
  alert(outerCon);
   }
  function $(id){
```

```
 return document.getElementById(id);
 }
 alert($('outer').innerHTML);
 alert($('outer').outerHTML);
</script>
</body>
</html>
```

108. JS 中的简单继承 call 方法!

```
//顶一个父母类，注意：类名都是首字母大写的哦!
 function Parent(name, money){
        this.name = name;
        this.money = money;
        this.info = function(){
            alert('姓名： '+this.name+' 钱： '+ this.money);
        }
    }
    //定义孩子类
    function Children(name){
        Parent.call(this, name); //继承 姓名属性，不要钱。
        this.info = function(){
            alert('姓名： '+this.name);
        }
    }
    //实例化类
    var per = new Parent('parent', 800000000000);
    var chi = new Children('child');
    per.info();
    chi.info();
```

109.

109. bind(), live(), delegate()的区别

bind： 绑定事件，对新添加的事件不起作用，方法用于将一个处理程序附加到每个匹配元素的事件上并返回 jQuery 对象。

live： 方法将一个事件处理程序附加到与当前选择器匹配的所有元素（包含现有的或将来添加的）的指定事件上并返回 jQuery 对象。

delegate： 方法基于一组特定的根元素将处理程序附加到匹配选择器的所有元素（现有的或将来的）的一个或多个事件上。

110. 看下列代码输出什么?

```
var foo = "11"+2-"1";
console.log(foo);
console.log(typeof foo);
```

执行完后 foo 的值为 111，foo 的类型为 Number。

111. 看下列代码,输出什么?

```
var a = new Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
```

执行完后输出结果为 2

112. 你如何优化自己的代码?

代码重用

避免全局变量（命名空间，封闭空间，模块化 mvc..）

拆分函数避免函数过于臃肿

注释

113. 请描述出下列代码运行的结果

```
function d(){
        console.log(this);
}
d();//输出 window 对象
```

114. 怎样实现两栏等高?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <div id="container" style="display: table;width: 100%;">
        <div id="left" style="background-color: red;display: table-cell;">
            内容<br/>
            内容<br/>
            内容<br/>
```

```
            内容<br/>
            内容<br/>
            内容<br/>
        </div>
        <div style="display:table-cell;"></div>
        <div id="right" style="background-color: blue;display: table-cell">
            内容
        </div>
    </div>
</body>
</html>
```

115. 使用 js 实现这样的效果：在文本域里输入文字时，当按下 enter 键时不换行，而是替换成“{{enter}}”,(只需要考虑在行尾按下 enter 键的情况).

```
<html>
<head>
    <script>
        function back(ele,event){
            event = event || window.event;
            if(event.keyCode==13){
```

```
            event.returnValue = false;
            ele.value+="{{enter}}"
            return false;
        }
    }
</script>
</head>
<body>
<textarea rows="3" cols="40" id="te"
onkeypress="back(this,event);"></textarea>
</body>
</html>
```

116. 以下代码中 end 字符串什么时候输出

```
var t=true;
setTimeout(function(){
    console.log(123);
    t=false;
    },1000);
while(t){}
console.log( 'end' );
```

永远不输出

117. specify(‘hello,world’)//=>’h,e,l,l,o,w,o,r,l,d’ 实现 specify 函数

```
function specify(str){
   var tempArray =
Array.prototype.filter.call(str,function(value,index,array){
      return value >= 'A' &&  value <= 'z' && value != "_";
   });
   return tempArray.join(",");
}

console.log(specify("hedd___df*(%$#a !!!)))))llo,Wo@@@r      ld"));
//h,e,l,l,o,W,o,r,l,d
```

118. 请将一个 URL 的 search 部分参数与值转换成一个 json 对象

119. 请用原生 js 实现 jquery 的 get\post 功能，以及跨域情况下

120. 请简要描述 web 前端性能需要考虑哪方面，你的优化思路是什么?

121. 、简述 readyonly 与 disabled 的区别

ReadOnly 和 Disabled 的作用是使用户不能够更改表单域中的内容.

但是二者还是有着一些区别的：

1、Readonly 只针对 input(text/password)和 textarea 有效，而 disabled 对于所有的表单元素有效，包括 select,radio,checkbox,button 等。

2、在表单元素使用了 disabled 后，我们将表单以 POST 或者 GET 的方式提交的话，这个元素的值不会被传递出去，而 readonly 会将该值传递出去

122. 写出 3 个使用 this 的典型应用

123. 请尽可能详尽的解释 ajax 的工作原理

Ajax 的工作原理相当于在用户和服务器之间加了一个中间层，使用户操作与服务器响应异步化。这样把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理，减轻服务器和带宽的负担，从而达到节约 ISP 的空间及带宽租用成本的目的。

简单来说通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 javascript 来操作 DOM 而更新页面。这其中最关键的一步就是从服务器获得请求数据。要清楚这个过程和原理，我们必须对 XMLHttpRequest 有所了解。

124. 、为什么扩展 javascript 内置对象不是好的做法?

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执行了数年，而浏览器的实现导致所有使用扩展原型的代码都崩溃了。。。

125. 什么是三元表达式？“三元”表示什么意思?

三元运算符:

三元如名字表示的三元运算符需要三个操作数。

语法是 条件？结果 1：结果 2;. 这里你把条件写在问号(?)的前面后面跟着用冒号(:)分隔的结果 1 和结果 2。满足条件时结果 1 否则结果 2。

126. 浏览器标准模式和怪异模式之间的区别是什么?

所谓的标准模式是指，浏览器按 W3C 标准解析执行代码；怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式

127. modulo(12,5)//2 实现满足这个结果的 modulo 函数

128. HTTP 协议中，GET 和 POST 有什么区别？分别适用什么场景？

129. HTTP 状态消息 200 302 304 403 404 500 分别表示什么

130. HTTP 协议中，header 信息里面，怎么控制页面失效时间（last-modified,cache-control,Expires 分别代表什么）

131. HTTP 雷锋议目前常用的有哪几个？KEEPALIVE 从哪个版本开始出现的?

132. 业界常用的优化 WEB 页面加载速度的方法（可以分别从页面元素展现，请求连接，css,js,服务器等方面介绍）

133. 列举常用的 web 页面开发，调试以及优化工具

134. 解释什么是 sql 注入，xss 漏洞

135. 如何判断一个 js 变量是数组类型

136. 请列举 js 数组类型中的常用方法

137. FF 与 IE 中如何阻止事件冒泡，如何获取事件对象，以及如何获取触发事件的元素

```
function stopPropagation(e) {
    e = e || window.event;
    if(e.stopPropagation) { //W3C 阻止冒泡方法
        e.stopPropagation();
    } else {
        e.cancelBubble = true; //IE 阻止冒泡方法
    }
}
document.getElementById('need_hide').onclick = function(e) {
    stopPropagation(e);
}
```

138.

138. 列举常用的 js 框架以及分别适用的领域

139. js 中如何实现一个 map

140. js 可否实现面向对象编程，如果可以如何实现 js 对象的继承

141. 约瑟夫环—已知 n 个人（以编号 1，2，3...分别表示）围坐在一张圆桌周围。从编号为 k 的人开始报数，数到 m 的那个人出列；他的下一个人又从 1 开始报数，数到 m 的那个人又出列；依此规律重复下去，直到圆桌周围的人全部出列。

142. 有 1 到 10w 这个 10w 个数，去除 2 个并打乱次序，如何找出那两个数?

143. 如何获取对象 a 拥有的所有属性（可枚举的、不可枚举的，不包括继承来的属性)

144. 有下面这样一段 HTML 结构，使用 css 实现这样的效果：

左边容器无论宽度如何变动，右边容器都能自适应填满父容器剩余的宽度。

145. 下面这段代码想要循环输出结果 01234，请问输出结果是否正确，如果不正确，请说明为什么，并修改循环内的代码使其输出正确结果

```
for(var i=0;i<5;++i){
    setTimeout(function(){
            console.log(i+’’);
        },100*i);
}
```

146. 以下哪些是 javascript 的全局函数：（ABC）

A. escape 函数可对字符串进行编码，这样就可以在所有的计算机上读取该字符串。ECMAScript v3 反对使用该方法，应用使用 decodeURI() 和 decodeURIComponent() 替代它。

B. parseFloat parseFloat() 函数可解析一个字符串，并返回一个浮点数。该函数指定字符串中的首个字符是否是数字。如果是，则对字符串进行解析，直到到达数字的末端为止，然后以数字返回该数字，而不是作为字符串。

C. eval 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

D. setTimeout

E. alert

147. 关于 IE 的 window 对象表述正确的有：（ACD）

A. window.opener 属性本身就是指向 window 对象

B. window.reload()方法可以用来刷新当前页面 应该是 location.reload 或者 window.location.reload

C. window.location=”a.html”和 window.location.href=”a.html”的作用都是把当前页面替换成 a.html 页面

D. 定义了全局变量 g；可以用 window.g 的方式来存取该变量

148. 下面正确的是 A

A: 跨域问题能通过 JsonP 方案解决 B：不同子域名间仅能通过修改 window.name 解决跨域 还可以通过 script 标签 src jsonp 等 h5 Java split 等

C：只有在 IE 中可通过 iframe 嵌套跨域 D：MediaQuery 属性是进行视频格式检测的属性是做响应式的

149. 错误的是 B

A: Ajax 本质是 XMLHttpRequest

B: 块元素实际占用的宽度与它的 width、border、padding 属性有关，与 background 无关

C: position 属性 absolute、fixed、---relative---会使文档脱标

D: float 属性 left 也会使 div 脱标

答案 C：relative 不会脱离文档流

150. 不用任何插件，如何实现一个 tab 栏切换?

151. 变量的命名规范以及命名推荐

变量，函数，方法：小写开头，以后的每个单词首字母大写 （驼峰）

构造函数，class：每个单词大写开头

基于实际情况，以动词，名词，谓词来命名。尽量言简意骇，以命名代替注释

152. 三种弹窗的单词以及三种弹窗的功能

1.alert

```
//弹出对话框并输出一段提示信息
    function ale() {
        //弹出一个对话框
        alert("提示信息！");

    }
```

2.confirm

```
    //弹出一个询问框，有确定和取消按钮
    function firm() {
        //利用对话框返回的值 （true 或者 false）
        if (confirm("你确定提交吗？")) {
            alert("点击了确定");
        }
        else {
            alert("点击了取消");
        }

    }
```

3.prompt

```
//弹出一个输入框，输入一段文字，可以提交
function prom() {
    var name = prompt("请输入您的名字", ""); //将输入的内容赋给变量 name，

    //这里需要注意的是，prompt 有两个参数，前面是提示的话，后面是当对话框出来后，在对话框里的默认值
    if (name)//如果返回的有内容
    {
        alert("欢迎您：" + name)
    }

}
```

153. console.log(8 | 1); 输出值是多少?

答案：9

154. 只允许使用 + - * / 和 Math.*，求一个函数 y = f(x, a, b);当 x > 100 时返回 a 的值，否则返回 b 的值，不能使用 if else 等条件语句，也不能使用|,?:,数组。

答案：

```
function f(x, a, b) {
```

```
    var temp = Math.ceil(Math.min(Math.max(x - 100, 0), 1));


    return a * temp + b * (1 - temp);
}
console.log(f(-10, 1, 2));
```

155. JavaScriptalert(0.4*0.2);结果是多少？和你预期的一样吗？如果不一样该如何处理?

有误差，应该比准确结果偏大。 一般我会将小数变为整数来处理。当前之前遇到这个问题时也上网查询发现有人用 try catch return 写了一个函数，

当然原理也是一致先转为整数再计算。

156. 一个 div，有几种方式得到这个 div 的 jQuery 对象？ 想直接获取这个 div 的 dom 对象，如何获取？dom 对象如何转化为 jQuery 对象?

$(“#nodesView”), $(“.aabbcc”),$(“#nodesView”)[0], $(“.aabbcc”)[0]

157. 、主流浏览器内核

IE trident 火狐 gecko 谷歌苹果 webkit Opera：Presto

158. 如何显示/隐藏一个 dom 元素？请用原生的 JavaScript 方法实现

159. jQuery 框架中$.ajax()的常用参数有哪些？写一个 post 请求并带有发送数据和返回数据的样例

async 是否异步

url 请求地址

contentType 发送信息至服务器时内容编码类型

data 发送到服务器的数据

dataType 预期服务器返回的数据类型

type 请求类型

success 请求成功回调函数

error 请求失败回调函数

```
$.ajax({
        url: "/jquery/test1.txt",
        type: 'post',
        data: {
            id: 1
        },
        success: function(data) {
            alert(data);
        }
    }
```

160. JavaScript 的循环语句有哪些?

For,for..in,while,do...while

161. 作用域-编译期执行期以及全局局部作用域问题

162. 闭包：下面这个 ul，如何点击每一列的时候 alert 其 index?

```
<ul id="test">
    <li>这是第一条</li>
    <li>这是第二条</li>
    <li>这是第三条</li>
</ul>
//js
 window.onload = function() {
        var lis = document.getElementById('test').children;
        for (var i = 0; i < lis.length; i++) {
            lis[i].onclick = (function(i) {
                return function() {
                    alert(i)
                };
            })(i);
        };
    }
```

163. 列出 3 条以上 ff 和 IE 的脚本兼容问题

(1) window.event :

表示当前的事件对象，IE 有这个对象，FF 没有，FF 通过给事件处理函数传递事件对象

(2) 获取事件源

IE 用 srcElement 获取事件源，而 FF 用 target 获取事件源

(3) 添加，去除事件

IE： element.attachEvent(“onclick”，function) element.detachEvent(“onclick”，function)

FF：element.addEventListener(“click”，function, true) element.removeEventListener(“click”，function, true)

(4) 获取标签的自定义属性

IE：div1.value 或 div1[“value”]

FF：可用 div1.getAttribute(“value”)

164. 如现在有一个效果，有显示用户头像、用户昵称、用户其他信息；当用户鼠标移到头像上时，会弹出用户的所有信息；如果是你，你会如何实现这个功能，请用代码实现?

(略)

提示：先写个 div 将用户信息放入，默认隐藏，当使用:hover 样式显示这个 div

165. 用正则表达式，写出由字母开头，其余由数字、字母、下划线组成的 6~30 的字符串?

^[a-zA-Z]{1}[\w]{5,29}$

166. 列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个 （10 分）

对象：Window document location screen history navigator

方法：Alert() confirm() prompt() open() close()

167. 在 Javascript 中什么是伪数组？如何将伪数组转化为标准数组?

答案：

伪数组（类数组）：无法直接调用数组方法或期望 length 属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的 argument 参数，还有像调用 getElementsByTagName,document.childNodes 之类的,它们都返回 NodeList 对象都属于伪数组。可以使用 Array.prototype.slice.call(fakeArray)将数组转化为真正的 Array 对象。

168. 写一个函数可以计算 sum(5,0,-5);输出 0; sum(1,2,3,4);输出 10;

```
function sum() {
    var result = 0;
    var arr = arguments;
    for (var i = 0; i < arr.length; i++) {
        var num = arguments[i];
```

```
        if (typeof num=='number') {
            result += num;
        };
    };
    return result;
}
```

169. 《正则》写出正确的正则表达式匹配固话号，区号 3-4 位，第一位为 0，中横线，7-8 位数字，中横线，3-4 位分机号格式的固话号

```
^[0]\d{2,3}\-\d{7,8}\-\d{3,4}$
```

170. 《算法》 一下 A,B 可任选一题作答，两题全答加分

A:农场买了一只羊，第一年是小羊，第二年底生一只，第三年不生，第四年底再生一只，第五年死掉。

B:写出代码对下列数组去重并从大到小排列{5,2,3,6,8,6,5,4,7,1,9}

```
function fn(arr){
    for (var i = 0; i < arr.length-1; i++) {
        for (var j = 0; j < arr.length-1-i; j++) {
            if(arr[j]<arr[j+1]){
                var temp = arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
```

```
        }


    }

}
for (i = 0; i < arr.length; i++) {
    var c=arr[i];
    for (var s = i+1; s < arr.length; s++) {
        if(arr[s]==c){
            //debugger;
            arr.splice(s,1);
            s--;
        }

    }
}

return arr;
```

```
}
console.log(fn([5,2,3,6,8,6,5,4,7,1,9]).toString());
```

171. 请写一个正则表达式：要求最短 6 位数，最长 20 位，阿拉伯数和英文字母（不区分大小写）组成

```
^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[a-zA-Z\d]{6,20}$
```

172. 统计 1 到 400 亿之间的自然数中含有多少个 1？比如 1-21 中，有 1、10、11、21 这四个自然数有 5 个 1

173. 删除与某个字符相邻且相同的字符，比如 fdaffdaaklfjklja 字符串处理之后成为“fdafdaklfjklja”

174. 请写出三种以上的 Firefox 有但，InternetExplorer 没有的属性或者函数

175. 请写出一个程序，在页面加载完成后动态创建一个 form 表单，并在里面添加一个 input 对象并给它任意赋值后义 post 方式提交到：http://127.0.0.1/save.php

```
window.onload=function(){
    var form=document.createElement("form");
```

```
    form.setAttribute("method", "post");
    form.setAttribute("action", "http://127.0.0.1/save.php");
    var input=document.createElement("input");
    form.appendChild(input);
    document.body.appendChild(form);
    input.value="cxc";
    form.submit();//提交表单
}
```

176. 用 JavaScript 实现冒泡排序。数据为 23、45、18、37、92、13、24

```
//升序算法
function sort(arr){
    for (var i = 0; i <arr.length; i++) {
        for (var j = 0; j <arr.length-i; j++) {
            if(arr[j]>arr[j+1]){
                var c=arr[j];//交换两个变量的位置
                arr[j]=arr[j+1];
```

```
                arr[j+1]=c;
            }
        };
    };
    return arr.toString();
}
console.log(sort([23,45,18,37,92,13,24]));
```

177. 前端代码优化的方法

```
var User = {
    count = 1,
    getCount: function () {
        return this.count;
    }
}
console.log(User.getCount());
var func = User.getCount;
console.log(func());
```

1 undefined（因为是 window 对象执行了 func 函数）；

178. 下列 JavaScript 代码执行后，依次 alert 的结果是

```
(function test(){
    var a=b=5;
    alert(typeof a);
    alert(typeof b);
})();
alert(typeof a);
alert(typeof b);
```

答案：number

number

undefined

number

179. 下列 JavaScript 代码执行后，iNum 的值是

```
var iNum = 0;
for(var i = 1; i< 10; i++){
   if(i % 5 == 0){
      continue;
  }
  iNum++;
```

```
}
```

答案：8

180. 输出结果是多少?

1) var a;

```
var b = a * 0;
if (b == b) {
        console.log(b * 2 + "2" - 0 + 4);
} else {
        console.log(!b * 2 + "2" - 0 + 4);
}
```

答案：26

2) <script>

```
    var a = 1;
</script>
<script>
var a;
var b = a * 0;
if (b == b) {
    console.log(b * 2 + "2" - 0 + 4);
} else {
    console.log(!b * 2 + "2" - 0 + 4);
```

```
}
</script>
```

答案：6

3） var t = 10;

```
function test(t){
    var t = t++;
}test(t);
console.log(t);
```

答案：10

4） var t = 10;

```
function test(test){
   var t = test++;
}test(t);
console.log(t);
```

答案：10

6） var t = 10;

```
function test(test){
    t = test++;
}test(t);
console.log(t);
```

答案：10

7） var t = 10;

```
function test(test){
    t = t + test;
    console.log(t);
    var t = 3;
}test(t);
console.log(t);
```

答案：NaN 10

8） var a;

```
var b = a / 0;
if (b == b) {
      console.log(b * 2 + "2" - 0 + 4);
} else {
      console.log(!b * 2 + "2" - 0 + 4);
}
```

答案：26

9）<script>

```
    var a = 1;
</script>
<script>
  var a;
  var b = a / 0;
  if (b == b) {
```

```
        console.log(b * 2 + "2" + 4);
    } else {
        console.log(!b * 2 + "2" + 4);
    }
</script>
```

答案：Infinity24

181. 用程序实现找到 html 中 id 名相同的元素?

```
<body>
    <form id='form1'>
        <div id='div1'></div>
        <div id='div2'></div>
        <div id='div3'></div>
        <div id='div4'></div>
        <div id='div5'></div>
        <div id='div3'>id 名重复的元素</div>
    </form>
</body>
```

```
var nodes=document.querySelectorAll("#form1>*");
for(var i=0,len=nodes.length;i<len;i++){
```

```
    var attr=nodes[i].getAttribute("id");
    var s=1;
    for(var j=i+1;j<len;j++){
        if(nodes[j].getAttribute("id")==attr){
            s++;
            alert("id 为：  "+attr+"的元素出现"+s+"次");
        }
    }
}
```

182.　下列 JavaScript 代码执行后，运行的结果是

```
<button id='btn'>点击我</button>
var btn = document.getElementById('btn');
var handler = {
    id: '_eventHandler',
    exec: function(){
        alert(this.id);
    }
}
```

```
btn.addEventListener('click', handler.exec);
```

答案：” btn”

183. 下列 JavaScript 代码执行后，依次 alert 的结果是

```
var obj = {proto: {a:1,b:2}};
function F(){};
F.prototype = obj.proto;
var f = new F();
obj.proto.c = 3;
obj.proto = {a:-1, b:-2};
alert(f.a);
alert(f.c);
delete F.prototype['a'];
alert(f.a);
alert(obj.proto.a);
```

答案：

1

3

undefined

-1

184. 下列 JavaScript 代码执行后的效果是

```
<ul id='list'>
    <li>item</li>
    <li>item</li>
    <li>item</li>
    <li>item</li>
    <li>item</li>
</ul>
var items = document.querySelectorAll('#list>li');
for(var i = 0;i < items.length; i++){
     setTimeout(function(){
          items[i].style.backgroundColor = '#fee';
    }, 5);
}
```

答案：报错，因为 i 一直等于 5，items[i]获取不到元素

185. 下列 JavaScript 代码执行后的 li 元素的数量是

```
<ul>
    <li>Item</li>
    <li></li>
    <li></li>
```

```
    <li>Item</li>
    <li>Item</li>
</ul>

var items = document.getElementsByTagName('li');
for(var i = 0; i< items.length; i++){
    if(items[i].innerHTML == ''){
        items[i].parentNode.removeChild(items[i]);
    }
}
```

186. 程序中捕获异常的方法?

```
window.error
try{}catch(){}finally{}
```

187. 将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的{$id}替换成 10，{$name}替换成 Tony （使用正则表达式）

答案:

```
'<tr><td>{$id}</td><td>{$id}_{$name}</td></tr>'
    .replace(/{\$id}/g,'10')
    .replace(/{\$name}/g,'Tony')
```

188. 给 String 对象添加一个方法，传入一个 string 类型的参数，然后将 string 的每个字符间价格空格返回，例如： addSpace(“hello world”) // -> ‘h e l l o ?w o r l d’

```
String.prototype.spacify = function(){
    return this.split('').join(' ');
};
```

189. 数组和字符串

```
<script lang="JavaScript" type="text/javascript">
  function outPut(s) {
     document.writeln(s);
  }
  var a = "lashou";
  var b = a;
  outPut(b);
  a = "拉手";
  outPut(a);
  outPut(b);
  var a_array = [1, 2, 3];
  var b_array = a_array;
  outPut(b_array);
```

```
    a_array[3] = 4;
    outPut(a_array);
    outPut(b_array);
</script>
```

输出结果：

答案：lashou 拉手 lashou 1,2,3 1,2,3,4 1,2,3,4

190. 下列控制台都输出什么

第 1 题：

```
function setName(){
    name="张三";
}
setName();
console.log(name);
```

答案："张三"

第 2 题：

```
//考点：1、变量声明提升 2、变量搜索机制
var a=1;
function test(){
    console.log(a);
    var a=1;
}
test();
```

答案：undefined

第 3 题：

```
var b=2;
function test2(){
    window.b=3;
    console.log(b);
}
test2();
```

答案：3

第 4 题：

```
c=5;//声明一个全局变量 c
function test3(){
    window.c=3;
    console.log(c);          //答案：undefined，原因：由于此时的 c 是一个局部变量 c，并且没有被赋值
    var c;
    console.log(window.c);//答案：3，原因：这里的 c 就是一个全局变量 c
}
test3();
```

第 5 题：

```
var arr = [];
arr[0]  = 'a';
```

```
arr[1]  = 'b';
arr[10] = 'c';
alert(arr.length); //答案：11
console.log(arr[5]);   //答案：undefined
```

第 6 题：

```
var a=1;
console.log(a++);          //答案：1
console.log(++a);          //答案：3
```

第 7 题：

```
console.log(null==undefined);    //答案：true
console.log("1"==1);      //答案：true，因为会将数字 1 先转换为字符串 1
console.log("1"===1);          //答案：false，因为数据类型不一致
```

第 8 题：

```
typeof 1;           "number"
typeof "hello";          "string"
typeof /[0-9]/;          "object"
typeof {};          "object"
typeof null;        "object"
typeof undefined;     "undefined"
typeof [1,2,3];          "object"
typeof function(){};   //"function"
```

第 9 题：

```
parseInt(3.14);                 //3
parseFloat("3asdf");          //3
parseInt("1.23abc456");
parseInt(true);//"true" NaN
```

第 10 题：

```
//考点：函数声明提前
function bar() {
    return foo;
    foo = 10;
    function foo() {}
    //var foo = 11;
}
alert(typeof bar());//"function"
```

第 11 题：考点：函数声明提前

```
var foo = 1;
function bar() {
        foo = 10;
        return;
        function foo() {}
}
bar();
```

```
alert(foo);//答案：1
```

第 12 题：

```
console.log(a);//是一个函数
var a = 3;
function a(){}
console.log(a);////3
```

第 13 题：

```
//考点：对 arguments 的操作
function foo(a) {
    arguments[0] = 2;
    alert(a);//答案：2，因为：a、arguments 是对实参的访问，b、通过 arguments[i]可以修改指定实参的值
}
foo(1);
```

第 14 题：

```
function foo(a) {
    alert(arguments.length);//答案：3，因为 arguments 是对实参的访问
}
foo(1, 2, 3);
```

第 15 题

```
bar();//报错
```

```
var foo = function bar(name) {
    console.log("hello"+name);
    console.log(bar);
};
//alert(typeof bar);
foo("world");//"hello"
console.log(bar);//undefined
console.log(foo.toString());
bar();//报错
```

第 16 题：以下执行会有什么输出

```
function test(){
    console.log("test 函数");
}
setTimeout(function(){
    console.log("定时器回调函数");
}, 0)
test();
```

结果：

test 函数

定时器回调函数

三、HTML5 CSS3

1. CSS3 有哪些新特性?

1. CSS3 实现圆角（border-radius），阴影（box-shadow），
2. 对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）
3.transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,倾斜
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，多栏布局
7. border-image

2. 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?

新特性：
1. 拖拽释放(Drag and drop) API
2. 语义化更好的内容标签（header,nav,footer,aside,article,section）
3. 音频、视频 API(audio,video)
4. 画布(Canvas) API
5. 地理(Geolocation) API
6. 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；
7. sessionStorage 的数据在浏览器关闭后自动删除
8. 表单控件，calendar、date、time、email、url、search
9. 新的技术 webworker, websocket, Geolocation
移除的元素：
1. 纯表现的元素：basefont，big，center，font, s，strike，tt，u；
2. 对可用性产生负面影响的元素：frame，frameset，noframes；
支持 HTML5 新标签：
1. 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

3. 本地存储（Local Storage ）和 cookies（储存在用户本地终端上的数据）之间的区别是什么?

Cookies:服务器和客户端都可以访问；大小只有 4KB 左右；有有效期，过期后将会删除；

本地存储：只有本地浏览器端可访问数据，服务器不能访问本地存储直到故意通过 POST 或者 GET 的通道发送到服务器；每个域 5MB；没有过期数据，它将保留知道用户从浏览器清除或者使用 Javascript 代码移除

4. 如何实现浏览器内多个标签页之间的通信?

调用 localstorge、cookies 等本地存储方式

5. 你如何对网站的文件和资源进行优化?

文件合并
文件最小化/文件压缩
使用 CDN 托管
缓存的使用

6. 什么是响应式设计?

它是关于网页制作的过程中让不同的设备有不同的尺寸和不同的功能。响应式设计是让所有的人能在这些设备上让网站运行正常

7. 新的 HTML5 文档类型和字符集是?

答：HTML5 文档类型：<!doctype html>
HTML5 使用的编码<meta charset=”UTF-8”>

8. HTML5 Canvas 元素有什么用?

答：Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在 HTML 上进行图形操作。

9. HTML5 存储类型有什么区别?

答：Media API、Text Track API、Application Cache API、User Interaction、Data Transfer API、Command API、Constraint Validation API、History API

10. 用 H5+CSS3 解决下导航栏最后一项掉下来的问题

11. CSS3 新增伪类有那些?

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。

12. 请用 CSS 实现：一个矩形内容，有投影，有圆角，hover 状态慢慢变透明。

css 属性的熟练程度和实践经验

13. 描述下 CSS3 里实现元素动画的方法

动画相关属性的熟悉程度

14. html5\CSS3 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，地理定位等功能的增加。

* 绘画 canvas 元素
  用于媒介回放的 video 和 audio 元素
  本地离线存储 localStorage 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
  CSS3 实现圆角，阴影，对文字加特效，增加了更多的 CSS 选择器 多背景 rgba
  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；

* 是 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]-->
```

15. 你怎么来实现页面设计图，你认为前端应该如何高质量完成工作？一个满屏 品 字布局 如何设计？

* 首先划分成头部、body、脚部；。。。。。
* 实现效果图是最基本的工作，精确到 2px；
  与设计师，产品经理的沟通和项目的参与
  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 HTML5。

16. 你能描述一下渐进增强和优雅降级之间的不同吗?

渐进增强 progressive enhancement：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。
优雅降级 graceful degradation：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

“优雅降级”观点

“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。而将那些被认为“过时”或有功能缺失的浏览器下的测试工作安排在开发周期的最后阶段，并把测试对象限定为主流浏览器（如 IE、Mozilla 等）的前一个版本。

在这种设计范例下，旧版的浏览器被认为仅能提供“简陋却无妨 (poor, but passable)” 的浏览体验。你可以做一些小的调整来适应某个特定的浏览器。但由于它们并非我们所关注的焦点，因此除了修复较大的错误之外，其它的差异将被直接忽略。

“渐进增强”观点

“渐进增强”观点则认为应关注于内容本身。

内容是我们建立网站的诱因。有的网站展示它，有的则收集它，有的寻求，有的操作，还有的网站甚至会包含以上的种种，但相同点是它们全都涉及到内容。这使得“渐进增强”成为一种更为合理的设计范例。这也是它立即被 Yahoo! 所采纳并用以构建其“分级式浏览器支持 (Graded Browser Support)”策略的原因所在。

那么问题了。现在产品经理看到 IE6,7,8 网页效果相对高版本现代浏览器少了很多圆角，阴影（CSS3），要求兼容（使用图片背景，放弃 CSS3），你会如何说服他?

17. 为什么利用多个域名来存储网站资源会更有效?

CDN 缓存更方便

突破浏览器并发限制

节约 cookie 带宽
节约主域名的连接数，优化页面响应速度
防止不必要的安全问题

18. 请谈一下你对网页标准和标准制定机构重要性的理解。

（无标准答案）网页标准和标准制定机构都是为了能让 web 发展的更‘健康’，开发者遵循统一的标准，降低开发难度，开发成本，SEO 也会更好做，也不会因为滥用代码导致各种 BUG、安全问题，最终提高网站易用性。

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

sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

web storage 和 cookie 的区别

Web Storage 的概念和 cookie 相似，区别是它是为了更大容量存储设计的。Cookie 的大小是受限的，并且每次你请求一个新的页面的时候 Cookie 都会被发送过去，这样无形中浪费了带宽，另外 cookie 还需要指定作用域，不可以跨域调用。
除此之外，Web Storage 拥有 setItem,getItem,removeItem,clear 等方法，不像 cookie 需要前端开发者自己封装 setCookie，getCookie。但是 Cookie 也是不可以或缺的：Cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在，而 Web Storage 仅仅是为了在本地“存储”数据而生。

20. 知道 css 有个 content 属性吗？有什么作用？有什么应用?

知道。css 的 content 属性专门应用在 before/after 伪元素上，用来插入生成内容。最常见的应用是利用伪类清除浮动。

```
//一种常见利用伪类清除浮动的代码

.clearfix:after {
    content:"."; //这里利用到了 content 属性
    display:block;
    height:0;
```

```
    visibility:hidden;
    clear:both; }
.clearfix {
    *zoom:1;
}
```

after 伪元素通过 content 在元素的后面生成了内容为一个点的块级素，再利用 clear:both 清除浮动。

那么问题继续还有，知道 css 计数器（序列数字字符自动递增）吗？如何通过 css content 属性实现 css 计数器？

答案：css 计数器是通过设置 counter-reset 、counter-increment 两个属性 、及 counter()/counters()一个方法配合 after / before 伪类实现。

21. 如何在 HTML5 页面中嵌入音频?

HTML 5 包含嵌入音频文件的标准方式，支持的格式包括 MP3、Wav 和 Ogg：

```
<audio controls>
  <source src="jamshed.mp3" type="audio/mpeg">
  Your browser does'nt support audio embedding feature.
</audio>
```

22. 如何在 HTML5 页面中嵌入视频?

和音频一样，HTML5 定义了嵌入视频的标准方法，支持的格式包括：MP4、WebM 和 Ogg：

```
<video width="450" height="340" controls>
  <source src="jamshed.mp4" type="video/mp4">
  Your browser does'nt support video embedding feature.
</video>
```

23. HTML5 引入什么新的表单属性?

Datalist datetime output keygen date month week time number range emailurl

24. CSS3 新增伪类有那些?

p:first-of-type 选择属于其父元素的首个 元素的每个 元素。
p:last-of-type 选择属于其父元素的最后 元素的每个 元素。
p:only-of-type 选择属于其父元素唯一的 元素的每个 元素。
p:only-child 选择属于其父元素的唯一子元素的每个 元素。
p:nth-child(2) 选择属于其父元素的第二个子元素的每个 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。

25. (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>
语义 HTML 具有以下特性：

文字包裹在元素中，用以反映内容。例如：
段落包含在 元素中。
顺序表包含在<ol>元素中。
从其他来源引用的大型文字块包含在<blockquote>元素中。
HTML 元素不能用作语义用途以外的其他目的。例如：
<h1>包含标题，但并非用于放大文本。
<blockquote>包含大段引述，但并非用于文本缩进。
空白段落元素（ ）并非用于跳行。
文本并不直接包含任何样式信息。例如：
不使用 <font> 或 等格式标记。
类或 ID 中不引用颜色或位置。

26. cookie 在浏览器和服务器间来回传递。 sessionStorage 和 localStorage 区别

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；

27. 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 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]-->

28. 如何区分： DOCTYPE 声明\新增的结构元素\功能元素

29. 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。

30. HTML5 的离线储存?

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。

31. 写出 HTML5 的文档声明方式

<DOCYPE html>

32.

32. HTML5 和 CSS3 的新标签

HTML5： nav, footer, header, section, hgroup, video, time, canvas, audio...
CSS3: RGBA, opacity, text-shadow, box-shadow, border-radius, border-image, border-color, transform...;

33.

33. 自己对标签语义化的理解

在我看来，语义化就是比如说一个段落， 那么我们就应该用 <p>标签来修饰，标题就应该用 <h?>标签等。符合文档语义的标签。

四、移动 web 开发

1、移动端常用类库及优缺点

知识面宽度，多多益善

2、Zepto 库和 JQ 区别

Zepto 相对 jQuery 更加轻量，主要用在移动端，jQuery 也有对应的 jQuerymobile 移动端框架

五、Ajax

1、Ajax 是什么? 如何创建一个 Ajax?

Ajax 并不算是一种新的技术，全称是 asychronous javascript and xml，可以说是已有技术的组合，主要用来实现客户端与服务器端的异步通信效果，实现页面的局部刷新，早期的浏览器并不能原生支持 ajax，可以使用隐藏帧（iframe）方式变相实现异步效果，后来的浏览器提供了对 ajax 的原生支持

使用 ajax 原生方式发送请求主要通过 XMLHttpRequest(标准浏览器)、ActiveXObject(IE 浏览器)对象实现异步通信效果

基本步骤:

```
var xhr =null;//创建对象
if(window.XMLHttpRequest){
    xhr = new XMLHttpRequest();
}else{

    xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
xhr.open(“方式”,”地址”,”标志位”);//初始化请求
xhr.setRequestHeader(“”,” ”);//设置 http 头信息
xhr.onreadystatechange =function(){}//指定回调函数
xhr.send();//发送请求
```

js 框架（jQuery/EXTJS 等）提供的 ajax API 对原生的 ajax 进行了封装，熟悉了基础理论，再学习别的框架就会得心应手，好多都是换汤不换药的内容

2、同步和异步的区别?

同步：阻塞的

-张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭

=浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

-张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃

=浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

3、如何解决跨域问题?

理解跨域的概念：协议、域名、端口都相同才同域，否则都是跨域

出于安全考虑，服务器不允许 ajax 跨域获取数据，但是可以跨域获取文件内容，所以基于这一点，可以动态创建 script 标签，使用标签的 src 属性访问 js 文件的形式获取 js 脚本，并且这个 js 脚本中的内容是**函数调用**，该函数调用的参数是服务器返回的数据，为了获取这里的参数数据，需要事先在页面中定义回调函数，在回调函数中处理服务器返回的数据，这就是解决跨域问题的主流解决方案

4、页面编码和被请求的资源编码如果不一致如何处理?

对于 ajax 请求传递的参数，如果是 get 请求方式，参数如果传递中文，在有些浏览器会乱码，不同的浏览器对参数编码的处理方式不同，所以对于 get 请求的参数需要使用 encodeURIComponent 函数对参数进行编码处理，后台开发语言都有相应的解码 api。对于 post 请求不需要进行编码

5、简述 ajax 的过程。

1. 创建 XMLHttpRequest 对象,也就是创建一个异步调用对象
2. 创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息
3. 设置响应 HTTP 请求状态变化的函数
4. 发送 HTTP 请求
5. 获取异步调用返回的数据
6. 使用 JavaScript 和 DOM 实现局部刷新

6、阐述一下异步加载。

1. 异步加载的方案： 动态插入 script 标签
2. 通过 ajax 去获取 js 代码，然后通过 eval 执行
3. script 标签上添加 defer 或者 async 属性
4. 创建并插入 iframe，让它异步执行 js

7、请解释一下 JavaScript 的同源策略。

同源策略是客户端脚本（尤其是 Javascript）的重要的安全度量标准。它最早出自 Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。所谓同源指

的是：协议，域名，端口相同，同源策略是一种安全协议，指一段脚本只能读取来自同一来源的窗口和文档的属性。

8、GET 和 POST 的区别，何时使用 POST?

GET：一般用于信息获取，使用 URL 传递参数，对所发送信息的数量也有限制，一般在 2000 个字符，有的浏览器是 8000 个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制

在以下情况中，请使用 POST 请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）
2. 向服务器发送大量数据（POST 没有数据量限制）
3. 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

9、ajax 是什么?ajax 的交互模型?同步和异步的区别?如何解决跨域问题?

1. 通过异步模式，提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax 在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

10、 Ajax 的最大的特点是什么。

Ajax 可以实现异步通信效果，实现页面局部刷新，带来更好的用户体验；按需获取数据，节约带宽资源；

11、ajax 的缺点

1、ajax 不支持浏览器 back 按钮。
2、安全问题 AJAX 暴露了与服务器交互的细节。
3、对搜索引擎的支持比较弱。
4、破坏了程序的异常机制。

12、ajax 请求的时候 get 和 post 方式的区别

get 一般用来进行查询操作，url 地址有长度限制，请求的参数都暴露在 url 地址当中，如果传递中文参数，需要自己进行编码操作，安全性较低。

post 请求方式主要用来提交数据，没有数据长度的限制，提交的数据内容存在于 http 请求体中，数据不会暴漏在 url 地址中。

13、解释 jsonp 的原理，以及为什么不是真正的 ajax

Jsonp 并不是一种数据格式，而 json 是一种数据格式，jsonp 是用来解决跨域获取数据的一种解决方案，具体是通过动态创建 script 标签，然后通过标签的 src 属性获取 js 文件中的 js 脚本，该脚本的内容是一个函数调用，参数就是服务器返回的数据，为了处理这些返回的数据，需要事先在页面定义好回调函数，本质上使用的并不是 ajax 技术

14、什么是 Ajax 和 JSON，它们的优缺点。

Ajax 是全称是 asynchronous JavaScript andXML，即异步 JavaScript 和 xml，用于在 Web 页面中实现异步数据交互，实现页面局部刷新。

优点：可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量，避免用户不断刷新或者跳转页面，提高用户体验

缺点：对搜索引擎不友好；要实现 ajax 下的前后退功能成本较大；可能造成请求数的增加跨域问题限制；

JSON 是一种轻量级的数据交换格式，ECMA 的一个子集

优点：轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）

15、http 常见的状态码有那些？分别代表是什么意思？

200 - 请求成功
301 - 资源（网页等）被永久转移到其它 URL
404 - 请求的资源（网页等）不存在
500 - 内部服务器错误

16、一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？

分为 4 个步骤：
1. 当发送一个 URL 请求时，不管这个 URL 是 Web 页面的 URL 还是 Web 页面上每个资源的 URL，浏览器都会开启一个线程来处理这个请求，同时在远程 DNS 服务器上启动一个 DNS 查询。这能使浏览器获得请求对应的 IP 地址。
2. 浏览器与远程 Web 服务器通过 TCP 三次握手协商来建立一个 TCP/IP 连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在 浏览器和服务器之

间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。
3. 一旦 TCP/IP 连接建立，浏览器会通过该连接向远程服务器发送 HTTP 的 GET 请求。远程服务器找到资源并使用 HTTP 响应返回该资源，值为 200 的 HTTP 响应状态表示一个正确的响应。
4. 此时，Web 服务器提供资源服务，客户端开始下载资源。

17、ajax 请求的时候 get 和 post 方式的区别

get 一般用来进行查询操作，url 地址有长度限制，请求的参数都暴露在 url 地址当中，如果传递中文参数，需要自己进行编码操作，安全性较低。

post 请求方式主要用来提交数据，没有数据长度的限制，提交的数据内容存在于 http 请求体中，数据不会暴漏在 url 地址中。

18、ajax 请求时，如何解释 json 数据

使用 eval() 或者 JSON.parse() 鉴于安全性考虑，推荐使用 JSON.parse()更靠谱，对数据的安全性更好。

19、.javascript 的本地对象，内置对象和宿主对象

本地对象为独立于宿主环境的 ECMAScript 提供的对象，包括 Array Object RegExp 等可以 new 实例化的对象

内置对象为 Gload，Math 等不可以实例化的(他们也是本地对象，内置对象是本地对象的一个子集)

宿主对象为所有的非本地对象，所有的 BOM 和 DOM 对象都是宿主对象，如浏览器自带的 document,window 等对象

20、为什么利用多个域名来存储网站资源会更有效?

确保用户在不同地区能用最快的速度打开网站，其中某个域名崩溃用户也能通过其他郁闷访问网站，并且不同的资源放到不同的服务器上有利于减轻单台服务器的压力。

21、请说出三种减低页面加载时间的方法

1、压缩 css、js 文件

2、合并 js、css 文件，减少 http 请求
3、外部 js、css 文件放在最底下
4、减少 dom 操作，尽可能用变量替代不必要的 dom 操作

22、HTTP 状态码都有那些。

200 OK　　//客户端请求成功
400 Bad Request　//客户端请求有语法错误，不能被服务器所理解
403 Forbidden　//服务器收到请求，但是拒绝提供服务
404 Not Found　//请求资源不存在，输入了错误的 URL
500 Internal Server Error //服务器发生不可预期的错误
503 Server Unavailable　//服务器当前不能处理客户端的请求，一段时间后可能恢复正常

六、JS 高级

1．JQuery 一个对象可以同时绑定多个事件，这是如何实现的?

jQuery 可以给一个对象同时绑定多个事件，低层实现方式是使用 addEventListner 或 attachEvent 兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

2．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?

Webkit 是浏览器引擎，包括 html 渲染和 js 解析功能，手机浏览器的主流内核，与之相对应的引擎有 Gecko（Mozilla Firefox 等使用）和 Trident（也称 MSHTML，IE 使用)。

对于浏览器的调试工具要熟练使用，主要是页面结构分析，后台请求信息查看，js 调试工具使用，熟练使用这些工具可以快速提高解决问题的效率

3．如何测试前端代码? 知道 BDD, TDD, Unit Test 么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

了解 BDD 行为驱动开发与 TDD 测试驱动开发已经单元测试相关概念，

4．前端 templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?

Web 模板引擎是为了使用户界面与业务数据（内容）分离而产生的，

Mustache 是一个 logic-less （轻逻辑）模板解析引擎，它的优势在于可以应用在 Javascript、PHP、Python、Perl 等多种编程语言中。

Underscore 封装了常用的 JavaScript 对象操作方法，用于提高开发效率。

Handlebars 是 JavaScript 一个语义模板库，通过对 view 和 data 的分离来快速构建 Web 模板。

5．简述一下 Handlebars 的基本用法?

没有用过的话说出它是干什么的即可

6．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?

学习技术不仅要会用，还有熟悉它的实现机制，这样在开发中遇到问题时才能更好的解决

7．用 js 实现千位分隔符?

原生 js 的熟练度，实践经验，实现思路

8．检测浏览器版本版本有哪些方式?

IE 与标准浏览器判断，IE 不同版本的判断，userAgent var ie = /*@cc_on !@*/false;

9．我们给一个 dom 同时绑定两个点击事件，一个用捕获，一个用冒泡，你来说下会执行几次事件，然后会先执行冒泡还是捕获

对两种事件模型的理解

10、实现一个函数 clone，可以对 JavaScript 中的 5 种主要的数据类型（包括 Number、String、Object、Array、Boolean）进行值复制

- 考察点 1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚
- 考察点 2：是否知道如何判断一个变量是什么类型的

- 考察点 3：递归算法的设计

```
// 方法一：
Object.prototype.clone = function(){
  var o = this.constructor === Array ? [] : {};
  for(var e in this){
    o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];
  }
  return o;
}
//方法二：
   /**
    * 克隆一个对象
    * @param Obj
    * @returns
    */
   function clone(Obj) {
      var buf;
      if (Obj instanceof Array) {
         buf = [];//创建一个空的数组
         var i = Obj.length;
         while (i--) {
            buf[i] = clone(Obj[i]);
         }
         return buf;
      }else if (Obj instanceof Object){
         buf = {};//创建一个空对象
         for (var k in Obj) { //为这个对象添加新的属性
            buf[k] = clone(Obj[k]);
         }
         return buf;
      }else{ //普通变量直接赋值
         return Obj;
      }
   }
```

-

11、如何消除一个数组里面重复的元素?

```
var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];
    function deRepeat(){
      var newArr=[];
      var obj={};
      var index=0;
      var l=arr.length;
      for(var i=0;i<l;i++){
        if(obj[arr[i]]==undefined)
         {
          obj[arr[i]]=1;
          newArr[index++]=arr[i];
         }
        else if(obj[arr[i]]==1)
         continue;
      }
      return newArr;
    }
    var newArr2=deRepeat(arr);
    alert(newArr2); //输出 1,2,3,4,5,6,9,25
```

12、小贤是一条可爱的小狗(Dog)，它的叫声很好听(wow)，每次看到主人的时候就会乖乖叫一声(yelp)。从这段描述可以得到以下对象：

```
function Dog() {
    this.wow = function() {
        alert('Wow');
    }
    this.yelp = function() {
        this.wow();
    }
}
```

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了(MadDog)，一看到人就会每隔半秒叫一声(wow)地不停叫唤(yelp)。请根据描述，按示例的形式用代码来实。（继承，原型，setInterval）

```
function MadDog() {
   this.yelp = function() {
       var self = this;
       setInterval(function() {
           self.wow();
       }, 500);
    }
}
MadDog.prototype = new Dog();
//for test
var dog = new Dog();
dog.yelp();
var madDog = new MadDog();
madDog.yelp();
```

13、下面这个 ul，如何点击每一列的时候 alert 其 index？（闭包）

```
<ul id=”test”>
<li>这是第一条</li>
<li>这是第二条</li>
<li>这是第三条</li>
</ul>
```

```
// 方法一：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++)
{
   lis[i].index=i;
   lis[i].onclick=function(){
      alert(this.index);
   };
}
//方法二：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++){
   lis[i].index=i;
   lis[i].onclick=(function(a){
      return function() {
         alert(a);
      }
   })(i);
}
```

14、编写一个 JavaScript 函数，输入指定类型的选择器(仅需支持 id，class，tagName 三种简单 CSS 选择器，无需兼容组合选择器)可以返回匹配的 DOM 节点，需考虑浏览器兼容性和性能。

```
/*** @param selector {String} 传入的 CSS 选择器。* @return {Array}*/
```

```
var query = function(selector) {
      var reg = /^(#)?(\.)?(\w+)$/img;
      var regResult = reg.exec(selector);
      var result = [];
      //如果是 id 选择器
      if(regResult[1]) {
      if(regResult[3]) {
            if(typeof document.querySelector === "function") {
                  result.push(document.querySelector(regResult[3]));
   }else {
    result.push(document.getElementById(regResult[3]));
   }
  }
  }
  //如果是 class 选择器
  else if(regResult[2]) {
    if(regResult[3]) {
     if(typeof document.getElementsByClassName === 'function') {
      var doms = document.getElementsByClassName(regResult[3]);
      if(doms) {
        result = converToArray(doms);
      }
     }
   //如果不支持 getElementsByClassName 函数
    else {
     var allDoms = document.getElementsByTagName("*") ;
     for(var i = 0, len = allDoms.length; i < len; i++) {
      if(allDoms[i].className.search(new RegExp(regResult[2])) > -1) {
       result.push(allDoms[i]);
       }
     }
    }
    }
    }
 //如果是标签选择器
 else if(regResult[3]) {
  var doms = document.getElementsByTagName(regResult[3].toLowerCase());
  if(doms) {
```

```
    result = converToArray(doms);
   }
  }
  return result;
  }
  function converToArray(nodes){
   var array = null;
   try{
    array = Array.prototype.slice.call(nodes,0);//针对非 IE 浏览器
    }catch(ex){
     array = new Array();
    for( var i = 0 ,len = nodes.length; i < len ; i++ ) {
     array.push(nodes[i])
    }
  }
  return array;
}
```

15、请评价以下代码并给出改进意见。

```
if(window.addEventListener){
    var addListener = function(el,type,listener,useCapture){
        el.addEventListener(type,listener,useCapture);
  };
}
else if(document.all){
    addListener = function(el,type,listener){
        el.attachEvent("on"+type,function(){
          listener.apply(el);
       });
    }
}
```

- 不应该在 if 和 else 语句中声明 addListener 函数，应该先声明；
- 不需要使用 window.addEventListener 或 document.all 来进行检测浏览器，应该使用能力检测；
- 由于 attachEvent 在 IE 中有 this 指向问题，所以调用它时需要处理一下

改进如下：

```
function addEvent(elem, type, handler){
    if(elem.addEventListener){
        elem.addEventListener(type, handler, false);
    }else if(elem.attachEvent){
        elem['temp' + type + handler] = handler;
        elem[type + handler] = function(){
        elem['temp' + type + handler].apply(elem);
    };
    elem.attachEvent('on' + type, elem[type + handler]);
  }else{
    elem['on' + type] = handler;
    }
}
```

16、给 String 对象添加一个方法，传入一个 string 类型的参数，然后将 string 的每个字符间价格空格返回，例如：

addSpace（“hello world”）// -> ‘h e l l o w o r l d’

```
String.prototype.spacify = function(){
  return this.split('').join(' ');
};
```

接着上述问题答案提问，1）直接在对象的原型上添加方法是否安全？尤其是在 Object 对象上。(这个我没能答出？希望知道的说一下。） 2）函数声明与函数表达式的区别？
答案：在 js 中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

17、定义一个 log 方法，让它可以代理 console.log 的方法。

可行的方法一：

```
function log(msg)  {
    console.log(msg);
}
log("hello world!") // hello world!
```

如果要传入多个参数呢？显然上面的方法不能满足要求，所以更好的方法是：

```
function log(){
    console.log.apply(console, arguments);
};
```

到此，追问 apply 和 call 方法的异同。

对于 apply 和 call 两者在作用上是相同的，即是调用一个对象的一个方法，以另一个对象替换当前对象。将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

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]) 。

18、在 Javascript 中什么是伪数组？如何将伪数组转化为标准数组?

伪数组（类数组）：无法直接调用数组方法或期望 length 属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的 argument 参数，还有像调用 getElementsByTagName,document.childNodes 之类的,它们都返回 NodeList 对象都属于伪数组。可以使用 Array.prototype.slice.call(fakeArray)将数组转化为真正的 Array 对象。

假设接第八题题干，我们要给每个 log 方法添加一个”(app)”前缀，比如’ hello world!’ ->’(app)hello world!’ 。方法如下：

```
function log(){
    var args = Array.prototype.slice.call(arguments); //为了使用 unshift 数组方法，将 argument 转化为真正的数组
    args.unshift('(app)');
    console.log.apply(console, args);
  };
```

19、对作用域上下文和 this 的理解，看下列代码：

```
var User = {
 count: 1,
 getCount: function() {
  return this.count;
 }
};
console.log(User.getCount()); // what?
var func = User.getCount;
console.log(func()); // what?
```

问两处 console 输出什么？为什么？

答案是 1 和 undefined。

func 是在 winodw 的上下文中被执行的，所以会访问不到 count 属性。

继续追问，那么如何确保 Uesr 总是能访问到 func 的上下文，即正确返回 1。正确的方法是使用 Function.prototype.bind。兼容各个浏览器完整代码如下：

```
Function.prototype.bind = Function.prototype.bind || function(context){
  var self = this;
  return function(){
    return self.apply(context, arguments);
  };
}
var func = User.getCount.bind(User);
console.log(func());
```

20、原生 JS 的 window.onload 与 Jquery 的 $(document).ready(function(){})有什么不同？如何用原生 JS 实现 Jq 的 ready 方法?

window.onload()方法是必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。

```
/*
 * 传递函数给 whenReady()
 * 当文档解析完毕且为操作准备就绪时，函数作为 document 的方法调用
 */
var whenReady = (function() {            //这个函数返回 whenReady()函数
  var funcs = [];          //当获得事件时，要运行的函数
  var ready = false;       //当触发事件处理程序时,切换为 true
  //当文档就绪时,调用事件处理程序
  function handler(e) {
    if(ready) return;     //确保事件处理程序只完整运行一次
    //如果发生 onreadystatechange 事件，但其状态不是 complete 的话,那么文档尚未准备好
    if(e.type === 'onreadystatechange' && document.readyState !== 'complete') {
      return;
    }
    //运行所有注册函数
    //注意每次都要计算 funcs.length
    //以防这些函数的调用可能会导致注册更多的函数
    for(var i=0; i<funcs.length; i++) {
      funcs[i].call(document);
    }
    //事件处理函数完整执行,切换 ready 状态, 并移除所有函数
    ready = true;
    funcs = null;
  }
  //为接收到的任何事件注册处理程序
  if(document.addEventListener) {
    document.addEventListener('DOMContentLoaded', handler, false);
    document.addEventListener('readystatechange', handler, false);          //IE9+
    window.addEventListener('load', handler, false);
  }else if(document.attachEvent) {
    document.attachEvent('onreadystatechange', handler);
    window.attachEvent('onload', handler);
  }
  //返回 whenReady()函数
  return function whenReady(fn) {
    if(ready) { fn.call(document); }
    else { funcs.push(fn); }
  }
```

```
})();
```

如果上述代码十分难懂，下面这个简化版：

```
function ready(fn){
  if(document.addEventListener) {//标准浏览器
    document.addEventListener('DOMContentLoaded', function() {
      //注销事件, 避免反复触发
      document.removeEventListener('DOMContentLoaded',arguments.callee, false);
      fn();//执行函数
    }, false);
  }else if(document.attachEvent) {//IE
    document.attachEvent('onreadystatechange', function() {
      if(document.readyState == 'complete') {
        document.detachEvent('onreadystatechange', arguments.callee);
        fn();//函数执行
      }
    });
  }
};
```

21、（设计题）想实现一个对页面某个节点的拖曳？如何做？（使用原生 JS）

回答出概念即可，下面是几个要点

1. 给需要拖拽的节点绑定 mousedown, mousemove, mouseup 事件
2. mousedown 事件触发后，开始拖拽
3. mousemove 时，需要通过 event.clientX 和 clientY 获取拖拽位置，并实时更新位置
4. mouseup 时，拖拽结束
5. 需要注意浏览器边界的情况

22、请实现如下功能


请实现下面功能（只实现tips组件部分）
1. 用户第一次进来时显示，同一天访问该页面不显示tip提示
2. 用户点击"我知道了"此后访问该页面不再显示tip提醒。
我的速卖通
产品管理
交易
站内信
网站设计
帐号设置
营销中心
数据纵横
商铺装修新功能上线！我知道了

```
 function setcookie(name,value,days){  //给 cookie 增加一个时间变量
        var exp = new Date();
        exp.setTime(exp.getTime() + days*24*60*60*1000); //设置过期时间为 days 天
        document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();
}
function getCookie(name){
        var result = "";
        var myCookie = ""+document.cookie+";";
        var searchName = "+name+"=";
        var startOfCookie = myCookie.indexOf(searchName);
        var endOfCookie;
        if(satrtOfCookie != -1){
                startOfcookie += searchName.length;
                endOfCookie = myCookie.indexOf(";",startOfCookie);
                result = (myCookie.substring(startOfCookie,endOfCookie));
        }
        return result;
}
(function(){
        var oTips = document.getElementById('tips');//假设 tips 的 id 为 tips
        var page = {
        check: function(){//检查 tips 的 cookie 是否存在并且允许显示
                var tips = getCookie('tips');
                if(!tips || tips == 'show') return true;//tips 的 cookie 不存在
                if(tips == "never_show_again") return false;
        },
        hideTip: function(bNever){
                if(bNever) setcookie('tips', 'never_show_again', 365);
                oTips.style.display = "none";//隐藏
        },
        showTip: function(){
        oTips.style.display = "inline";//显示，假设 tips 为行级元素
        },
        init: function(){
                var _this = this;
                if(this.check()){
                _this.showTip();
                setcookie('tips', 'show', 1);
```

```
        }
        oTips.onclick = function(){
                _this.hideTip(true);
        };
        }
        };
  page.init();
})();
```

23、说出以下函数的作用是？空白区域应该填写什么？

```
//define
(function(window){
    function fn(str){
        this.str=str;
    }

    fn.prototype.format = function(){
        var arg = _______;
        return this.str.replace(_____,function(a,b){
            return arg[b]||"";
      });
    }
    window.fn = fn;
})(window);

//use
(function(){
    var t = new fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');
    console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));
})();
```

答案：访函数的作用是使用 format 函数将函数的参数替换掉{0}这样的内容，返回一个格式化后的结果：

第一个空是：arguments

第二个空是：/\{(\d+)\}/ig

24. Javascript 作用链域?

理解变量和函数的访问范围和生命周期，全局作用域与局部作用域的区别，JavaScript 中没有块作用域，函数的嵌套形成不同层次的作用域，嵌套的层次形成链式形

式，通过作用域链查找属性的规则需要深入理解。

25. 谈谈 This 对象的理解。

理解不同形式的函数调用方式下的 this 指向，理解事件函数、定时函数中的 this 指向，函数的调用形式决定了 this 的指向。

26. 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 个步骤，一次解析成 js 语句，一次执行）

27. 关于事件，IE 与火狐的事件机制有什么区别？ 如何阻止冒泡?

[1].在 IE 中,事件对象是作为一个全局变量来保存和维护的.所有的浏览器事件,不管是用户触发的，还是其他事件,都会更新 window.event 对象.所以在代码中，只要调用 window.event 就可以获取事件对象， 再 event.srcElement 就可以取得触发事件的元素进行进一步处理.

[2].在 FireFox 中，事件对象却不是全局对象，一般情况下，是现场发生，现场使用，FireFox 把事件对象自动传给事件处理程序.

关于事件的兼容性处理要熟练掌握，事件对象具体哪些属性存在兼容性问题，IE 与标准事件模型事件冒泡与事件捕获的支持要理解

28. 什么是闭包（closure），为什么要用它?

简单的理解是函数的嵌套形成闭包，闭包包括函数本身已经它的外部作用域

使用闭包可以形成独立的空间，延长变量的生命周期，报存中间状态值

29、javascript 代码中的"use strict";是什么意思？使用它区别是什么?

意思是使用严格模式，使用严格模式，一些不规范的语法将不再支持

30、如何判断一个对象是否属于某个类?

Instanceof　constructor

31、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 。

32、用原生 JavaScript 的实现过什么功能吗?

主要考察原生 js 的实践经验

33、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

34、对 JSON 的了解?

轻量级数据交互格式，可以形成复杂的嵌套格式，解析非常方便

35、js 延迟加载的方式有哪些?

方案一：<script>标签的 async="async"属性（详细参见：script 标签的 async 属性）

方案二：<script>标签的 defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用 AJAX 得到脚本内容，然后通过 eval_r(xmlhttp.responseText)

来运行脚本)

方案五：iframe 方式

36、模块化开发怎么做?

理解模块化开发模式：浏览器端 requirejs，seajs；服务器端 nodejs；ES6 模块化；fis、webpack 等前端整体模块化解决方案；grunt、gulp 等前端工作流的使用

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

理解这两种规范的差异，主要通过 requirejs 与 seajs 的对比，理解模块的定义与引用方式的差异以及这两种规范的设计原则

38、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？）

核心是 js 的加载模块，通过正则匹配模块以及模块的依赖关系，保证文件加载的先后顺序，根据文件的路径对加载过的文件做了缓存

39、让你自己设计实现一个 requireJS，你会怎么做?

核心是实现 js 的加载模块，维护 js 的依赖关系，控制好文件加载的先后顺序

40、谈一谈你对 ECMAScript6 的了解?

ES6 新的语法糖，类，模块化等新特性

41、ECMAScript6 怎么写 class 么，为什么会出现 class 这种东西?

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
  toString() {
     return '('+this.x+', '+this.y+')';
  }
}
```

42、异步加载的方式有哪些?

方案一：<script>标签的 async="async"属性（详细参见：script 标签的 async 属性）

方案二：<script>标签的 defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用 AJAX 得到脚本内容，然后通过 eval_r(xmlhttp.responseText)来运行脚本）

方案五：iframe 方式

43、documen.write 和 innerHTML 的区别?

document.write 是重写整个 document, 写入内容是字符串的 html

innerHTML 是 HTMLElement 的属性，是一个元素的内部 html 内容

44、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_x()　//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

45、call() 和 .apply() 的含义和区别?

apply 的参数是数组形式，call 的参数是单个的值，除此之外在使用上没有差别，重点理解这两个函数调用的 this 改变

46、数组和对象有哪些原生方法，列举一下?

Array.concat( ) 连接数组

Array.join( ) 将数组元素连接起来以构建一个字符串

Array.length 数组的大小

Array.pop( ) 删除并返回数组的最后一个元素

Array.push( ) 给数组添加元素

Array.reverse( ) 颠倒数组中元素的顺序

Array.shift( ) 将元素移出数组

Array.slice( ) 返回数组的一部分

Array.sort( ) 对数组元素进行排序

Array.splice( ) 插入、删除或替换数组的元素

Array.toLocaleString( ) 把数组转换成局部字符串

Array.toString( ) 将数组转换成一个字符串

Array.unshift( ) 在数组头部插入一个元素

Object.hasOwnProperty( ) 检查属性是否被继承

Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

Object.propertyIsEnumerable( ) 是否可以通过 for/in 循环看到属性

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

47、JS 怎么实现一个类。怎么实例化这个类

严格来讲 js 中并没有类的概念，不过 js 中的函数可以作为构造函数来使用，通过 new 来实例化，其实函数本身也是一个对象。

48、JavaScript 中的作用域与变量声明提升?

理解 JavaScript 的预解析机制，js 的运行主要分两个阶段：js 的预解析和运行，预解析阶段所有的变量声明和函数定义都会提前，但是变量的赋值不会提前

49、如何编写高性能的 Javascript?

使用 DocumentFragment 优化多次 append

通过模板元素 clone，替代 createElement

使用一次 innerHTML 赋值代替构建 dom 元素

使用 firstChild 和 nextSibling 代替 childNodes 遍历 dom 元素

使用 Array 做为 StringBuffer，代替字符串拼接的操作

将循环控制量保存到局部变量

顺序无关的遍历时，用 while 替代 for

将条件分支，按可能性顺序从高到低排列

在同一条件子的多（>2）条件分支时，使用 switch 优于 if

使用三目运算符替代条件分支

需要不断执行的时候，优先考虑使用 setInterval

50、那些操作会造成内存泄漏?

闭包，循环

51、javascript 对象的几种创建方式?

1. 工厂模式
2. 构造函数模式
3. 原型模式
4. 混合构造函数和原型模式
5. 动态原型模式
6. 寄生构造函数模式
7. 稳妥构造函数模式

52、javascript 继承的 6 种方法?

1. 原型链继承
2. 借用构造函数继承
3. 组合继承(原型+借用构造)
4. 原型式继承
5. 寄生式继承
6. 寄生组合式继承

53、eval 是做什么的?

1. 它的功能是把对应的字符串解析成 JS 代码并运行
2. 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）

54、JavaScript 原型，原型链？有什么特点?

1. 原型对象也是普通的对象，是对象一个自带隐式的 __proto__ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 null 的话，我们就称之为原型链
2. 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

55、事件、IE 与火狐的事件机制有什么区别？ 如何阻止冒泡?

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为
2. 事件处理机制：IE 是事件冒泡、firefox 同时支持两种事件模型，也就是：捕获型事件和冒泡型事件
3. ev.stopPropagation();
注意旧 ie 的方法：ev.cancelBubble = true;

56、简述一下 Sass、Less，且说明区别?

他们是动态的样式语言，是 CSS 预处理器,CSS 上的一种抽象层。他们是一种特殊的语法/语言而编译成 CSS。
变量符不一样，less 是@，而 Sass 是$;
Sass 支持条件语句，可以使用 if{}else{},for{}循环等等。而 Less 不支持;
Sass 是基于 Ruby 的，是在服务端处理的，而 Less 是需要引入 less.js 来处理 Less 代码输

出 Css 到浏览器

57、关于 javascript 中 apply()和 call()方法的区别?

相同点:两个方法产生的作用是完全一样的
不同点:方法传递的参数不同
Object.call(this,obj1,obj2,obj3)
Object.apply(this,arguments)
apply()接收两个参数，一个是函数运行的作用域(this)，另一个是参数数组。
call()方法第一个参数与 apply()方法相同，但传递给函数的参数必须列举出来。

58、简述一下 JS 中的闭包?

闭包用的多的两个作用：读取函数内部的变量值；让这些变量值始终保存着(在内存中)。
同时需要注意的是：闭包慎用，不滥用，不乱用，由于函数内部的变量都被保存在内存中，会导致内存消耗大。

59、说说你对 this 的理解?

在 JavaScript 中，this 通常指向的是我们正在执行的函数本身，或者是，指向该函数所属的对象。
全局的 this → 指向的是 Window
函数中的 this → 指向的是函数所在的对象
对象中的 this → 指向其本身

60、分别阐述 split(),slice(),splice(),join()?

join()用于把数组中的所有元素拼接起来放入一个字符串。所带的参数为分割字符串的分隔符，默认是以逗号分开。归属于 Array
split()即把字符串分离开，以数组方式存储。归属于 Stringstring
slice() 方法可从已有的数组中返回选定的元素。该方法并不会修改数组，而是返回一个子数组。如果想删除数组中的一段元素，应该使用方法 Array.splice()
splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。返回的是含有被删除的元素的数组。

61、事件委托是什么?

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行!

62、如何阻止事件冒泡和默认事件?

阻止浏览器的默认行为

window.event?window.event.returnValue=false:e.preventDefault();

停止事件冒泡

window.event?window.event.cancelBubble=true:e.stopPropagation();

原生 JavaScript 中，return false;只阻止默认行为，不阻止冒泡，jQuery 中的 return false;既阻止默认行为，又阻止冒泡

63、添加 删除 替换 插入到某个接点的方法?

obj.appendChidl()
obj.removeChild()
obj.replaceChild()
obj.innersetBefore()

64、你用过 require.js 吗？它有什么特性?

（1）实现 js 文件的异步加载，避免网页失去响应；
（2）管理模块之间的依赖性，便于代码的编写和维护。

65、谈一下 JS 中的递归函数，并且用递归简单实现阶乘?

递归即是程序在执行过程中不断调用自身的编程技巧，当然也必须要有一个明确的结束条件，不然就会陷入死循环。

66、请用正则表达式写一个简单的邮箱验证。

/^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$/;

67、简述一下你对 web 性能优化的方案?

1、尽量减少 HTTP 请求
2、使用浏览器缓存
3、使用压缩组件
4、图片、JS 的预载入
5、将脚本放在底部
6、将样式文件放在页面顶部
7、使用外部的 JS 和 CSS
8、精简代码

68、在 JS 中有哪些会被隐式转换为 false

Undefined、null、关键字 false、NaN、零、空字符串

69、定时器 setInterval 有一个有名函数 fn1，setInterval（fn1,500）与 setInterval（fn1(),500）有什么区别?

第一个是重复执行每 500 毫秒执行一次，后面一个只执行一次。

70、外部 JS 文件出现中文字符，会出现什么问题，怎么解决?

会出现乱码，加 charset=”GB2312”；

71、谈谈浏览器的内核，并且说一下什么是内核?

Trident (['traɪd(ə)nt])--IE , Gecko (['gekəʊ])--Firefox, Presto (['prestəʊ])--opera,webkit—谷歌和 Safari
浏览器内核又可以分成两部分：渲染引擎和 JS 引擎。它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。JS 引擎则是解析 Javascript 语言，执行 javascript 语言来实现网页的动态效果。

72、JavaScript 原型，原型链？有什么特点?

* 原型对象也是普通的对象，是对象一个自带隐式的 __proto__ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 null 的话，我们就称之为原型链。
* 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。
* JavaScript 的数据对象有那些属性值?
    writable：这个属性的值是否可以改。
    configurable：这个属性的配置是否可以删除，修改。
    enumerable：这个属性是否能在 for...in 循环中遍历出来或在 Object.keys 中列举出来。
    value：属性值。
* 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。

```
 function clone(proto) {
     function Dummy() { }
     Dummy.prototype = proto;
     Dummy.prototype.constructor = Dummy;
     return new Dummy(); //等价于 Object.create(Person);
 }
      function object(old) {
       function F() {};
       F.prototype = old;
       return new F();
      }
   var newObj = object(oldObject);
```

73、写一个通用的事件侦听器函数

```
`// event(事件)工具集，来源：https://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;
    }
};
```

74、事件、IE 与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE 是事件冒泡、火狐是 事件捕获；
3. ev.stopPropagation();

75、什么是闭包（closure），为什么要用?

执行 say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在.使得 Javascript 的垃圾回收机制 GC 不会收回 say667()所占用的资源，因为 say667()的内部函数的执行需要依赖 say667()中的变量。这是对闭包作用的非常直白的描述.

```
 function say667() {
   // Local variable that ends up within closure
   var num = 666;
   var sayAlert = function() { alert(num); }
   num++;
   return sayAlert;
}
var sayAlert = say667();
sayAlert()//执行结果应该弹出的 667
```

76、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');
}
```

77、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);
```

78、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
{'age':'12', 'name':'back'}

79、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

80、模块化怎么做?

立即执行函数,不暴露私有成员

```
var module1 = (function(){
    var _count = 0;
    var m1 = function(){
        //...
    };
    var m2 = function(){
        //...
    };
    return {
        m1 : m1,
        m2 : m2
    };
})();
```

81、异步加载的方式

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

82、告诉我答案是多少?

```
(function(x){
  delete x;
```

```
    alert(x);
})(1+5);
```

函数参数无法 delete 删除，delete 只能删除通过 for in 访问的属性。
当然，删除失败也不会报错，所以代码运行会弹出“1”。

83、JS 中的 call()和 apply()方法的区别?

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1)，所以运行结果为：alert(4);
注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

```
function add(a,b){
    alert(a+b);
}
function sub(a,b){
    alert(a-b);
}
add.call(sub,3,1);
```

84、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

85、jquery 中如何将数组转化为 json 字符串，然后再转化回来?

jQuery 中没有提供这个功能，所以你需要先编写两个 jQuery 的扩展：

```
    $.fn.stringifyArray = function(array) {

        return JSON.stringify(array)
    }
    $.fn.parseArray = function(array) {
        return JSON.parse(array)
    }
```

然后调用：

```
    $("").stringifyArray(array)
```

86、JavaScript 中的作用域与变量声明提升?

其他部分
（HTTP、正则、优化、重构、响应式、移动端、团队协作、SEO、UED、职业生涯）
*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++) {}

87、前端开发的优化问题（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。

88、http 状态码有那些？分别代表是什么意思?

100-199 用于指定客户端应相应的某些动作。
200-299 用于表示请求成功。
300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。
400-499 用于指出客户端的错误。
400 语义有误，当前请求无法被服务器理解。
401 当前请求需要用户验证
403 服务器已经理解请求，但是拒绝执行它。
500-599 用于支持服务器错误。
503 – 服务不可用

89、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）

要熟悉前后端的通信流程，最好把动态网站的背后细节也介绍一遍

七、流行框架

1、JQuery 的源码看过吗？能不能简单概况一下它的实现原理?

考察学习知识的态度，是否仅仅是停留在使用层面，要知其然知其所以然

2、jQuery.fn 的 init 方法返回的 this 指的是什么对象？为什么要返回 this?

this执行init构造函数自身，其实就是jQuery实例对象，返回this是为了实现jQuery的链式操作

3．jquery 中如何将数组转化为 json 字符串，然后再转化回来?

$.parseJSON('{"name":"John"}');

4．jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝?

递归赋值

5．jquery.extend 与 jquery.fn.extend 的区别?

Jquery.extend 用来扩展 jQuery 对象本身；jquery.fn.extend 用来扩展 jQuery 实例

6、谈一下 Jquery 中的 bind(),live(),delegate(),on()的区别?

7、JQuery 一个对象可以同时绑定多个事件，这是如何实现的?

可以同时绑定多个事件，低层实现原理是使用 addEventListner 与 attachEvent 兼容

处理做事件注册

10.Jquery 与 jQuery UI 有啥区别?

jQuery 是操作 dom 的框架，jQueryUI 是基于 jQuery 做的一个 UI 组件库

11.jQuery 和 Zepto 的区别？各自的使用场景?

jQuery 主要用于 pc 端，当然有对应的 jQuerymobile 用于移动端，zepto 比 jQuery 更加小巧，主要用于移动端

12.针对 jQuery 的优化方法?

优先使用 ID 选择器

在 class 前使用 tag(标签名)

给选择器一个上下文

慎用 .live()方法（应该说尽量不要使用）

使用 data()方法存储临时变量

13.Zepto 的点透问题如何解决?

点透主要是由于两个 div 重合，例如：一个 div 调用 show()，一个 div 调用 hide()；这个时候当点击上面的 div 的时候就会影响到下面的那个 div；

解决办法主要有 2 种：

1.github 上有一个叫做 fastclick 的库，它也能规避移动设备上 click 事件的延迟响应，https://github.com/ftlabs/fastclick

将它用 script 标签引入页面（该库支持 AMD，于是你也可以按照 AMD 规范，用诸如 require.js 的模块加载器引入），并且在 dom ready 时初始化在 body 上，

2.根据分析，如果不引入其它类库，也不想自己按照上述 fastclcik 的思路再开发一套东西，需要 1.一个优先于下面的“divClickUnder”捕获的事件；2.并且通过这个事件阻止掉默认行为（下面的“divClickUnder”对 click 事件的捕获，在 ios 的 safari，

click 的捕获被认为和滚屏、点击输入框弹起键盘等一样，是一种浏览器默认行为，即可以被 event.preventDefault()阻止的行为)。

14、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?

知识面的宽度，流行框架要多多熟悉

15、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?

Underscore 的熟悉程度

16、使用过 angular 吗? angular 中的过滤器是干什么用的

在表达式中转换数据 姓名为 {{ lastName | uppercase }}
currency，是什么过滤器——格式化数字为货币格式，单位是$符。

八、移动 APP 开发

1、移动端最小触控区域是多大?

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。）

九、NodeJs

68. 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。

69. 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?

至少给出自己的思路（url-hash,可以使用已有的一些框架 history.js 等）

70. Node.js 的适用场景?

1)、实时应用：如在线聊天，实时通知推送等等（如 socket.io）

2)、分布式应用：通过高效的并行 I/O 使用已有的数据

3)、工具类应用：海量的工具，小到前端压缩部署（如 grunt），大到桌面图形界面应用程序

4)、游戏类应用：游戏领域对实时和并发有很高的要求（如网易的 pomelo 框架）

5)、利用稳定接口提升 Web 渲染能力

6)、前后端编程语言环境统一：前端开发人员可以非常快速地切入到服务器端的开发（如著名的纯 Javascript 全栈式 MEAN 架构)

71. (如果会用 node)知道 route, middleware, cluster, nodemon, pm2, server-side rendering 么?

Nodejs 相关概念的理解程度

72. 解释一下 Backbone 的 MVC 实现方式?

流行的 MVC 架构模式

73. 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?

熟悉前后端通信相关知识

74. 对 Node 的优点和缺点提出了自己的看法?

优点：
1. 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。
2. 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：
1. 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。
2. 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（第三方库现在已经很丰富了，所以这个缺点可以说不存在了）。

十、前端概括性问题

75. 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?

使用率较高的框架有 jQuery、YUI、Prototype、Dojo、Ext.js、Mootools 等。尤其是 jQuery，超过 91%。

轻量级框架有 Modernizr、underscore.js、backbone.js、Raphael.js 等。(理解这些框架的功能、性能、设计原理)

前端开发工具：Sublime Text 、Eclipse、Notepad、Firebug、HttpWatch、Yslow。

开发过的插件：城市选择插件，汽车型号选择插件、幻灯片插件。弹出层。(写过开源程序，加载器，js 引擎更好)

76. 对 BFC 规范的理解?

Formatting Context：指页面中的一个渲染区域，并且拥有一套渲染规则，他决定了

其子元素如何定位，以及与其他元素的相互关系和作用。

77. 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）

78. WEB 应用从服务器主动推送 Data 到客户端有那些方式?

html5 websoket
WebSocket 通过 Flash
XHR 长时间连接
XHR Multipart Streaming
不可见的 Iframe
<script>标签的长时间连接(可跨域)

79. 加班的看法

加班就像借钱，原则应当是------救急不救穷

80. 平时如何管理你的项目，如何设计突发大规模并发架构?

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

81. 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）

82. 你说你热爱前端，那么应该 WEB 行业的发展很关注吧？ 说说最近最流行的一些东西吧？

Node.js、Mongodb、npm、MVVM、MEAN、react、angularjs

83. 你有了解我们公司吗？说说你的认识？

因为我想去阿里，所以我针对阿里的说
最羡慕就是在双十一购物节，350.19 亿元，每分钟支付 79 万笔。海量数据，居然无一漏单、无一故障。太厉害了。

84. 移动端（比如：Android IOS）怎么做好用户体验？

融入自己的设计理念，注重用户体验，选择合适的技术

85. 你所知道的页面性能优化方法有那些？

压缩、合并，减少请求，代码层析优化。。。

86. 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？

知识面宽度，最好熟悉一些后台语言，比如 php，展现出自己的技术两点

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

88. 谈谈你认为怎样做能使项目做的更好？

考虑问题的深入，不仅仅停留在完成任务上，要精益求精

89. 你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？

表现出对前端的认同与兴趣，关注相关技术前沿

90. php 中下面哪个函数可以打开一个文件，以对文件进行读和写操作？

A.fget(); B.file_open(); **C.fopen();** D.open_file();

91. php 中 rmdir 可以直接删除文件夹吗？该目录必须是空的，而且要有相应的权限--来自 api

A.任何文件夹都可以删除　　B.空文件夹可以删除

C.有权限的任何文件夹都可以删除　D.有权限的空文件夹可以删除

92. phpinset 和 empty 的区别，举例说明

1、empty 函数

用途：检测变量是否为空

判断：如果 var 是非空或非零的值，则 empty() 返回 FALSE。换句话说，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE。注意：empty() 只检测变量，检测任何非变量的东西都将导致解析错误。换句话说，后边的语句将不会起作用;

2、isset 函数

用途：检测变量是否设置

判断：检测变量是否设置，并且不是 NULL。如果已经使用 unset() 释放了一个变量之后，它将不再是 isset()。若使用 isset() 测试一个被设置成 NULL 的变量，将返回 FALSE。同时要注意的是一个 NULL 字节（"\0"）并不等同于 PHP 的 NULL 常数。

93. php 中$_SERVER 变量中如何得到当前执行脚本路劲

```
__FILE__: ========> G:\web\test\t2\dir.php
__DIR__: ========> G:\web\test\t2
dirname(__FILE__): ========> G:\web\test\t2
$_SERVER["PHP_SELF"]: ========> /test/t2/dir.php
$_SERVER["SCRIPT_NAME"]: ========> /test/t2/dir.php
$_SERVER["SCRIPT_FILENAME"]: ========> G:/web/test/t2/dir.php
$_SERVER["DOCUMENT_ROOT"]: ========> G:/web
getcwd(): ========> G:\web\test\t2
```

94. 写一个 php 函数，要求两个日期字符串的天数差，如 2012-02-05~2012-03-06 的日期差数

95. 一个衣柜中放了许多杂乱的衬衫，如果让你去整理一下，使得更容易找到你想要的衣服；你会怎么做？请写出你的做法和思路?

96. 如何优化网页加载速度?

1.减少 css，js 文件数量及大小(减少重复性代码，代码重复利用)，压缩 CSS 和 Js 代码
2.图片的大小
3.把 css 样式表放置顶部，把 js 放置页面底部
4.减少 http 请求数
5.使用外部 Js 和 CSS

97. 工作流程，你怎么来实现页面设计图，你认为前端应该如何高质量完成工作?

熟悉相关设计规范，自己总结的一些经验

98. 介绍项目经验、合作开发、独立开发。

团队协作，个人能力。实践经验

99. 开发过程中遇到困难，如何解决。

考察解决问题的能力

100. 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。

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. 瀑布流布局或者流式布局是否有了解

4. HTML5 都有哪些新的 API?

5. 都用过什么代码调试工具?

6. 是否有接触过或者了解过重构。

7.你遇到过比较难的技术问题是？你是如何解决的?

Vue

一、对于 MVVM 的理解？

MVVM 是 Model-View-ViewModel 的缩写。

Model 代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑。

View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来。

ViewModel 监听模型数据的改变和控制视图行为、处理用户交互，简单理解就是一个同步 View 和 Model 的对象，连接 Model 和 View。

在 MVVM 架构下，View 和 Model 之间并没有直接的联系，而是通过 ViewModel 进行交互，Model 和 ViewModel 之间的交互是双向的， 因此 View 数据的变化会同步到 Model 中，而 Model 数据的变化也会立即反应到 View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而 View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作 DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

二、Vue 的生命周期

beforeCreate（创建前） 在数据观测和初始化事件还未开始

created（创建后） 完成数据观测，属性和方法的运算，初始化事件，$el 属性还没有显示出来

beforeMount（载入前） 在挂载开始之前被调用，相关的 render 函数首次被调用。实例已完成以下的配置：编译模板，把 data 里面的数据和模板生成 html。注意此时还没有挂载 html 到页面上。

mounted（载入后） 在 el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用。实例已完成以下的配置：用上面编译好的 html 内容替换 el 属性指向的 DOM 对象。完成模板中的 html 渲染到 html 页面中。此过程中进行 ajax 交互。

beforeUpdate（更新前） 在数据更新之前调用，发生在虚拟 DOM 重新渲染和打补丁之前。可以在该钩子中进一步地更改状态，不会触发附加的重渲染过程。

updated（更新后） 在由于数据更改导致的虚拟 DOM 重新渲染和打补丁之后调用。调用时，组件 DOM 已经更新，所以可以执行依赖于 DOM 的操作。然而在大多数情况下，应该避免在此期间更改状态，因为这可能会导致更新无限循环。该钩子在服务器端渲染期间不被调用。

beforeDestroy（销毁前） 在实例销毁之前调用。实例仍然完全可用。

destroyed（销毁后） 在实例销毁之后调用。调用后，所有的事件监听器会被移除，所有的子实例也会被销毁。该钩子在服务器端渲染期间不被调用。

1.什么是 vue 生命周期？

答： Vue 实例从创建到销毁的过程，就是生命周期。从开始创建、初始化数据、编译模板、挂载 Dom→渲染、更新→渲染、销毁等一系列过程，称之为 Vue 的生命周期。

2.vue 生命周期的作用是什么？

答：它的生命周期中有多个事件钩子，让我们在控制整个 Vue 实例的过程时更容易形成好的逻辑。

3.vue 生命周期总共有几个阶段？

答：它可以总共分为 8 个阶段：创建前/后, 载入前/后,更新前/后,销毁前/销毁后。

4.第一次页面加载会触发哪几个钩子？

答：会触发 下面这几个 beforeCreate, created, beforeMount, mounted 。

5.DOM 渲染在 哪个周期中就已经完成？

答：DOM 渲染在 mounted 中就已经完成了。

三、 Vue 实现数据双向绑定的原理：Object.defineProperty（）

vue 实现数据双向绑定主要是：采**用数据劫持结合发布者-订阅者模式**的方式，通过 **Object.defineProperty**（）来劫持各个属性的 setter，getter，在数据变动时发布消息给订阅者，触发相应监听回调。当把一个普通 Javascript 对象传给 Vue 实例来

作为它的 data 选项时，Vue 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。用户看不到 getter/setter，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。

vue 的数据双向绑定 将 MVVM 作为数据绑定的入口，整合 Observer，Compile 和 Watcher 三者，通过 Observer 来监听自己的 model 的数据变化，通过 Compile 来解析编译模板指令（vue 中是用来解析 {{}}），最终利用 watcher 搭起 observer 和 Compile 之间的通信桥梁，达到数据变化 —>视图更新；视图交互变化（input）—>数据 model 变更双向绑定效果。

js实现简单的双向绑定

```
<body>
    <div id="app">
    <input type="text" id="txt">
    <p id="show"></p>
</div>
</body>
<script type="text/javascript">
    var obj = {}
    Object.defineProperty(obj, 'txt', {
        get: function () {
            return obj
        },
        set: function (newValue) {
            document.getElementById('txt').value = newValue
            document.getElementById('show').innerHTML = newValue
        }
    })
    document.addEventListener('keyup', function (e) {
        obj.txt = e.target.value
    })
</script>
```

四、Vue 组件间的参数传递

1.父组件与子组件传值

父组件传给子组件：子组件通过 props 方法接受数据;

子组件传给父组件：$emit 方法传递参数

2.非父子组件间的数据传递，兄弟组件传值

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。项目比较小时，用这个比较合适。（虽然也有不少人推荐直接用 VUEX，具体来说看需求咯。技术只是手段，目的达到才是王道。）

五、Vue 的路由实现：hash 模式 和 history 模式

hash 模式：在浏览器中符号“#”，#以及#后面的字符称之为 hash，用 window.location.hash 读取；

特点：hash 虽然在 URL 中，但不被包括在 HTTP 请求中；用来指导浏览器动作，对服务端安全无用，hash 不会重加载页面。

hash 模式下，仅 hash 符号之前的内容会被包含在请求中，

如 http://www.xxx.com，因此对于后端来说，即使没有做到对路由的全覆盖，也不会返回 404 错误。

history 模式：history 采用 HTML5 的新特性；且提供了两个新方法：pushState（），replaceState（）可以对浏览器历史记录栈进行修改，以及 popState 事件的监听到状态变更。

history 模式下，前端的 URL 必须和实际向后端发起请求的 URL 一致，

如 http://www.xxx.com/items/id。后端如果缺少对 /items/id 的路由处理，将返回 404 错误。**Vue-Router 官网里如此描述：**“不过这种模式要玩好，还需要后台配置支持……所以呢，你要在服务端增加一个覆盖所有情况的候选资源：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面，这个页面就是你 app 依赖的页面。”

六、Vue 与 Angular 以及 React 的区别？

（版本在不断更新，以下的区别有可能不是很正确。我工作中只用到 vue，对 angular 和 react 不怎么熟）

1.与 AngularJS 的区别

相同点：

都支持指令：内置指令和自定义指令；都支持过滤器：内置过滤器和自定义过滤

器；都支持双向数据绑定；都不支持低端浏览器。

不同点：
AngularJS 的学习成本高，比如增加了 Dependency Injection 特性，而 Vue.js 本身提供的 API 都比较简单、直观；在性能上，AngularJS 依赖对数据做脏检查，所以 Watcher 越多越慢；Vue.js 使用基于依赖追踪的观察并且使用异步队列更新，所有的数据都是独立触发的。

2.与 React 的区别
相同点：
React 采用特殊的 JSX 语法，Vue.js 在组件开发中也推崇编写.vue 特殊文件格式，对文件内容都有一些约定，两者都需要编译后使用；中心思想相同：一切都是组件，组件实例之间可以嵌套；都提供合理的钩子函数，可以让开发者定制化地去处理需求；都不内置列数 AJAX，Route 等功能到核心包，而是以插件的方式加载；在组件开发中都支持 mixins 的特性。
不同点：
React 采用的 Virtual DOM 会对渲染出来的结果做脏检查；Vue.js 在模板中提供了指令，过滤器等，可以非常方便，快捷地操作 Virtual DOM。

七、vue 路由的钩子函数

首页可以控制导航跳转，beforeEach，afterEach 等，一般用于页面 title 的修改。一些需要登录才能调整页面的重定向功能。

beforeEach 主要有 3 个参数 to，from，next：

to：route 即将进入的目标路由对象，

from：route 当前导航正要离开的路由

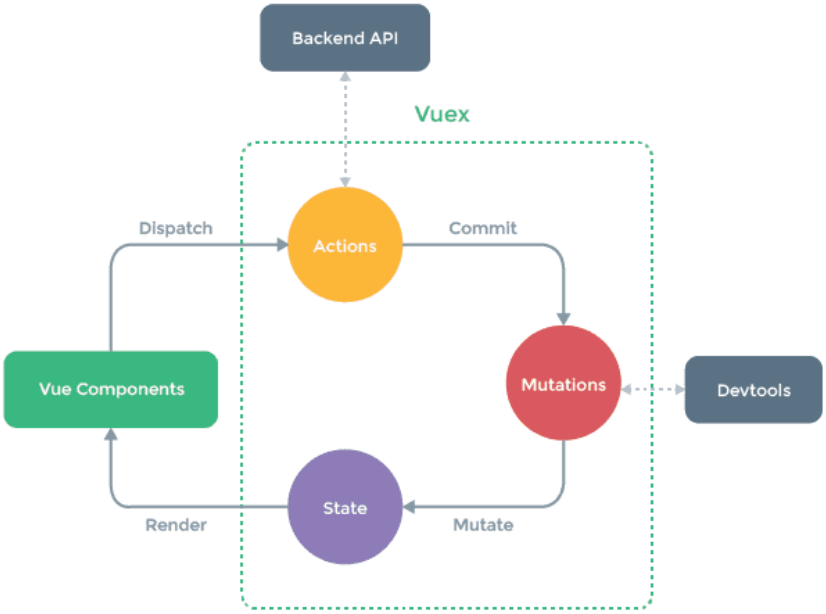
next：function 一定要调用该方法 resolve 这个钩子。执行效果依赖 next 方法的调用参数。可以控制网页的跳转。

八、vuex 是什么？怎么使用？哪种功能场景使用它？

只用来读取的状态集中放在 store 中； 改变状态的方式是提交 mutations，这是个同步的事物； 异步逻辑应该封装在 action 中。

在 main.js 引入 store，注入。新建了一个目录 store，….. export 。

场景有：单页应用中，组件之间的状态、音乐播放、登录状态、加入购物车



state

Vuex 使用单一状态树,即每个应用将仅仅包含一个 store 实例，但单一状态树和模块化并不冲突。存放的数据状态，不可以直接修改里面的数据。

mutations

mutations 定义的方法动态修改 Vuex 的 store 中的状态或数据。

getters

类似 vue 的计算属性，主要用来过滤一些数据。

action

actions 可以理解为通过将 mutations 里面处里数据的方法变成可异步的处理数据的方法，简单的说就是异步操作数据。view 层通过 store.dispath 来分发 action。

```
const store = new Vuex.Store({ //store实例
      state: {
         count: 0
             },
      mutations: {
         increment (state) {
          state.count++
         }
          },
      actions: {
         increment (context) {
          context.commit('increment')
   }
 }
})
```

modules

项目特别复杂的时候，可以让每一个模块拥有自己的 state、mutation、action、getters,使得结构非常清晰，方便管理。

```
const moduleA = {
  state: { ... },
  mutations: { ... },
  actions: { ... },
  getters: { ... }
 }
const moduleB = {
  state: { ... },
  mutations: { ... },
  actions: { ... }
 }

const store = new Vuex.Store({
  modules: {
    a: moduleA,
    b: moduleB
})
```

九、vue-cli 如何新增自定义指令？

1.创建局部指令

```
var app = new Vue({
    el: '#app',
    data: {
    },
    // 创建指令(可以多个)
    directives: {
        // 指令名称
        dir1: {
            inserted(el) {
                // 指令中第一个参数是当前使用指令的DOM
                console.log(el);
                console.log(arguments);
                // 对DOM进行操作
                el.style.width = '200px';
                el.style.height = '200px';
                el.style.background = '#000';
            }
        }
    }
})
```

2.全局指令

```
Vue.directive('dir2', {
    inserted(el) {
        console.log(el);
    }
})
```

3.指令的使用

```
<div id="app">
    <div v-dir1></div>
    <div v-dir2></div>
</div>
```

十、vue 如何自定义一个过滤器？

html 代码：

```
<div id="app">
    <input type="text" v-model="msg" />
    {{msg| capitalize }}
</div>
```

JS 代码：

```
var vm=new Vue({
    el:"#app",
    data:{
        msg:''
    },
    filters: {
      capitalize: function (value) {
        if (!value) return ''
        value = value.toString()
        return value.charAt(0).toUpperCase() + value.slice(1)
      }
    }
})
```

全局定义过滤器

```
Vue.filter('capitalize', function (value) {
  if (!value) return ''
  value = value.toString()
  return value.charAt(0).toUpperCase() + value.slice(1)
})
```

过滤器接收表达式的值 (msg) 作为第一个参数。capitalize 过滤器将会收到 msg 的值作为第一个参数。

十一、对 keep-alive 的了解？

keep-alive 是 Vue 内置的一个组件，可以使被包含的组件保留状态，或避免重新渲染。

在 vue 2.1.0 版本之后，keep-alive 新加入了两个属性: include(包含的组件缓存) 与 exclude(排除的组件不缓存，优先级大于 include) 。

使用方法

```
<keep-alive include='include_components' exclude='exclude_components'>
  <component>
    <!-- 该组件是否缓存取决于include和exclude属性 -->
  </component>
</keep-alive>
```

参数解释
include - 字符串或正则表达式，只有名称匹配的组件会被缓存
exclude - 字符串或正则表达式，任何名称匹配的组件都不会被缓存

include 和 exclude 的属性允许组件有条件地缓存。二者都可以用“，”分隔字符串、正则表达式、数组。当使用正则或者是数组时，要记得使用 v-bind 。

使用示例

```
<!-- 逗号分隔字符串，只有组件a与b被缓存。 -->
<keep-alive include="a,b">
  <component></component>
</keep-alive>

<!-- 正则表达式 （需要使用 v-bind，符合匹配规则的都会被缓存） -->
<keep-alive :include="/a|b/">
  <component></component>
</keep-alive>

<!-- Array （需要使用 v-bind，被包含的都会被缓存） -->
<keep-alive :include="['a', 'b']">
  <component></component>
</keep-alive>
```

十二、一句话就能回答的面试题

1.css 只在当前组件起作用

答：在 style 标签中写入 **scoped** 即可 例如：<style scoped></style>

2.v-if 和 v-show 区别

答：v-if 按照条件是否渲染，v-show 是 display 的 block 或 none；

3.$route 和$router 的区别

答：$route 是“路由信息对象”，包括 path，params，hash，query，fullPath，matched，name 等路由信息参数。而$router 是“路由实例”对象包括了路由的跳转方法，钩子函数等。

4.vue.js 的两个核心是什么？

答：数据驱动、组件系统

5.vue 几种常用的指令

答：v-for 、 v-if 、v-bind、v-on、v-show、v-else

6.vue 常用的修饰符？

答：.prevent: 提交事件不再重载页面；.stop: 阻止单击事件冒泡；.self: 当事件发生在该元素本身而不是子元素的时候会触发；.capture: 事件侦听，事件发生的时候会调用

7.v-on 可以绑定多个方法吗？

答：可以

8.vue 中 key 值的作用？

答：当 Vue.js 用 v-for 正在更新已渲染过的元素列表时，它默认用“就地复用”策略。如果数据项的顺序被改变，Vue 将不会移动 DOM 元素来匹配数据项的顺序，而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素。key 的作用主要是为了高效的更新虚拟 DOM。

9.什么是 vue 的计算属性？

答：在模板中放入太多的逻辑会让模板过重且难以维护，在需要对数据进行复杂处理，且可能多次使用的情况下，尽量采取计算属性的方式。好处：

①使得数据处理结构清晰；

②依赖于数据，数据更新，处理结果自动更新；

③计算属性内部 this 指向 vm 实例；

④在 template 调用时，直接写计算属性名即可；

⑤常用的是 getter 方法，获取数据，也可以使用 set 方法改变数据；

⑥相较于 methods，不管依赖的数据变不变，methods 都会重新计算，但是依赖数据不变的时候 computed 从缓存中获取，不会重新计算。

10.vue 等单页面应用及其优缺点

答：优点：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，核心是一个响应的数据绑定系统。MVVM、数据驱动、组件化、轻量、简洁、高效、快速、模块友好。
缺点：不支持低版本的浏览器，最低只支持到 IE9；不利于 SEO 的优化（如果要支持 SEO，建议通过服务端来进行渲染组件）；第一次加载首页耗时相对长一些；不可以使用浏览器的导航按钮需要自行实现前进、后退。

React

React 中 keys 的作用是什么?

Keys 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

```
render () {
  return (
    <ul>
      {this.state.todoItems.map(({item, key}) => {
        return <li key={key}>{item}</li>
      })}
    </ul>
  )
}
```

在开发过程中，我们需要保证某个元素的 key 在其同级元素中具有唯一性。在 React Diff 算法中 React 会借助元素的 Key 值来判断该元素是新近创建的还是被移动而来的元素，从而减少不必要的元素重渲染。此外，React 还需要借助 Key 值来判断元素与本地状态的关联关系，因此我们绝不可忽视转换函数中 Key 的重要性。

调用 setState 之后发生了什么?

在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对

高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

react 生命周期函数

- 初始化阶段：
    - getDefaultProps:获取实例的默认属性
    - getInitialState:获取每个实例的初始化状态
    - componentWillMount：组件即将被装载、渲染到页面上
    - render:组件在这里生成虚拟的 DOM 节点
    - componentDidMount:组件真正在被装载之后
- 运行中状态：
    - componentWillReceiveProps:组件将要接收到属性的时候调用
    - shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）
    - componentWillUpdate:组件即将更新不能修改属性和状态
    - render:组件重新描绘
    - componentDidUpdate:组件已经更新
- 销毁阶段：
    - componentWillUnmount:组件即将销毁

shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘

dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

为什么虚拟 dom 会提高性能?(必考)

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

react diff 原理（常考，大厂必考）

- 把树形结构按照层级分解，只比较同级元素。
- 给列表结构的每个单元添加唯一的 key 属性，方便比较。
- React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
- 合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty. 到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.
- 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

React 中 refs 的作用是什么?

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回：

```
class CustomForm extends Component {
  handleSubmit = () => {
    console.log("Input Value: ", this.input.value)
  }
  render () {
    return (
      <form onSubmit={this.handleSubmit}>
        <input
          type='text'
          ref={(input) => this.input = input} />
        <button type='submit'>Submit</button>
      </form>
    )
  }
}
```

上述代码中的 input 域包含了一个 ref 属性，该属性声明的回调函数会接收 input 对应的 DOM 元素，我们将其绑定到 this 指针以便在其他的类函数中使用。另外值得一提的是，refs 并不是类组件的专属，函数式组件同样能够利用闭包暂存其值：

```
function CustomForm ({handleSubmit}) {
  let inputElement
  return (
    <form onSubmit={() => handleSubmit(inputElement.value)}>
      <input
        type='text'
        ref={(input) => inputElement = input} />
      <button type='submit'>Submit</button>
    </form>
  )
}
```

如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的？

```
<Twitter username='tylermcginnis33'>
  {(user) => user === null
    ? <Loading />
    : <Badge info={user} />}
</Twitter>
```

```
import React, { Component, PropTypes } from 'react'
import fetchUser from 'twitter'
// fetchUser take in a username returns a promise
// which will resolve with that username's data.
class Twitter extends Component {
  // finish this
}
```

如果你还不熟悉回调渲染模式（Render Callback Pattern），这个代码可能看起来有点怪。这种模式中，组件会接收某个函数作为其子组件，然后在渲染函数中以 props.children 进行调用：

```
import React, { Component, PropTypes } from 'react'
import fetchUser from 'twitter'
class Twitter extends Component {
  state = {
    user: null,
  }
  static propTypes = {
    username: PropTypes.string.isRequired,
  }
  componentDidMount () {
    fetchUser(this.props.username)
      .then((user) => this.setState({user}))
  }
  render () {
    return this.props.children(this.state.user)
  }
}
```

这种模式的优势在于将父组件与子组件解耦和，父组件可以直接访问子组件的内部状态而不需要再通过 Props 传递，这样父组件能够更为方便地控制子组件展示的 UI 界面。譬如产品经理让我们将原本展示的 Badge 替换为 Profile，我们可以轻易地修改下回调函数即可：

```
<Twitter username='tylermcginnis33'>
  {(user) => user === null
    ? <Loading />
    : <Profile info={user} />}
</Twitter>
```

展示组件(Presentational component)和容器组件(Container component)之间有何不同

- 展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。
- 容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

类组件(Class component)和函数式组件(Functional component)之间有何不同

- 类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态

- 当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

(组件的)状态(state)和属性(props)之间有何不同

- State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。
- Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

何为受控组件(controlled component)

在 HTML 中，类似 `<input>`, `<textarea>` 和 `<select>` 这样的表单元素会维护自身的状态，并基于用户的输入来更新。当用户提交表单时，前面提到的元素的值将随表单一起被发送。但在 React 中会有些不同，包含表单元素的组件将会在 state 中追踪输入的值，并且每次调用回调函数时，如 onChange 会更新 state，重新渲染组件。一个输入表单元素，它的值通过 React 的这种方式来控制，这样的元素就被称为"受控元素"。

何为高阶组件(higher order component)

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

为什么建议传递给 setState 的参数是一个 callback 而不是一个对象

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

除了在构造函数中绑定 this，还有其它方式吗

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。

(在构造函数中)调用 super(props) 的目的是什么

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

应该在 React 组件的何处发起 Ajax 请求

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

描述事件在 React 中的处理方式。

为了解决跨浏览器兼容性问题，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。React 将使用单个事件监听器监听顶层的所有事件。这对于性能是有好处的，这也意味着在更新 DOM 时，React 不需要担心跟踪事件监听器。

createElement 和 cloneElement 有什么区别?

React.createElement():JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名。如 div、span，或者 React 组件。第二个参数为传入的属性。第三个以及之后的参数，皆作为组件的子组件。

```
React.createElement(
    type,
    [props],
    [...children]
)
```

React.cloneElement()与 React.createElement()相似，不同的是它传入的第一个参数是一个 React 元素，而不是标签名或组件。新添加的属性会并入原有的属性，传入到返回的新元素中，而就的子元素奖杯替换。

```
React.cloneElement(
  element,
  [props],
  [...children]
)
```

React 中有三种构建组件的方式

React.createClass()、ES6 class 和无状态函数。

react 组件的划分业务组件技术组件?

- 根据组件的职责通常把组件分为 UI 组件和容器组件。
- UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。
- 两者通过 React-Redux 提供 connect 方法联系起来。

简述 flux 思想

Flux 的最大特点，就是数据的"单向流动"。

1. 用户访问 View
2. View 发出用户的 Action
3. Dispatcher 收到 Action，要求 Store 进行相应的更新
4. Store 更新后，发出一个"change"事件
5. View 收到"change"事件后，更新页面

React 项目用过什么脚手架（本题是开放性题目）

creat-react-app Yeoman 等

了解 redux 么，说一下 redux 把

- redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰

- 新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们

redux 有什么缺点

- 一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。
- 当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。