面试题集

1. HTML 和 CSS

2. HTML5 CSS3

3. JS 基础

4. Ajax

5. JS 高级

6. 流行框架

7. 移动 APP 开发

8. NodeJs

9. 前端概括性问题

一、HTML 和 CSS

10. 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？

Trident 内核:IE 系列
Gecko 内核:Firefox
Webkit 内核:Safari
Blink 内核：是基于 Webkit 内核的子项目,使用的浏览器有：
Chrome/opera 等除 IE、Firefox、Safari 之外的几乎所有浏览器
几乎所有国产双内核浏览器（Trident/Blink）如 360、猎豹、qq、百度等

11. 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？

文档声明。
<!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）
IE 下如不书写文档声明会使用怪异模式解析网页导致一系列 CSS 兼容性问题。

12. div+css 的布局较 table 布局有什么优点？

正常场景一般都适用 div+CSS 布局，优点：
结构与样式分离
代码语义性好
更符合 HTML 标准规范
SEO 友好

Table 布局的适用场景：
某种原因不方便加载外部 CSS 的场景，例如邮件正文，此时用 table 布局可以在无 css 情况下保持页面布局正常。

13. 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。
em:表现为斜体，语义表示强调
strong:表现为粗体，语义为强烈语气，强调程度超过 em

14. 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。

15. 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

16. 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。

17. 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。

18. 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。

19. 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%

20. 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。

21. 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。

22. 你如何理解 HTML 结构的语义化？

HTML 结构语义化：
更符合 W3C 统一的规范标准，是技术趋势。
没有样式时浏览器的默认样式也能让页面结构很清晰。

对功能障碍用户友好。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。
对其他非主流终端设备友好。例如机顶盒、PDA、各种移动终端。
对 SEO 友好。

23. 谈谈以前端角度出发做好 SEO 需要考虑什么？

搜索引擎主要以:
外链数量和质量
网页内容和结构
来决定某关键字下的网页搜索排名。

前端应该注意网页结构和内容方面的情况：
Meta 标签优化
主要包括主题（Title），网站描述(Description)。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。
符合 W3C 规范的语义性标签的使用。

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。

24. 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

25. CSS 都有哪些选择器？

一、基本选择器

1.	*	通用元素选择器，匹配任何元素
2.	E	标签选择器，匹配所有使用 E 标签的元素
3.	.info	class 选择器，匹配所有 class 属性中包含 info 的元素
4.	#footer	id 选择器，匹配所有 id 属性等于 footer 的元素

二、多元素的组合选择器

5.	E,F	多元素选择器，同时匹配所有 E 元素或 F 元素，E 和 F 之间用逗号分隔
6.	E F	后代元素选择器，匹配所有属于 E 元素后代的 F 元素，E 和 F 之间用空格分隔
7.	E > F	子元素选择器，匹配所有 E 元素的子元素 F
8.	E + F	毗邻元素选择器，匹配所有紧随 E 元素之后的同级元素 F

三、CSS 2.1 属性选择器

9. E[att] 匹配所有具有 att 属性的 E 元素，不考虑它的值。（注意：E 在此处可以省略，比如"[cheacked]"。以下同。）

10. E[att=val]匹配所有 att 属性等于"val"的 E 元素

11. E[att~=val] 匹配所有 att 属性具有多个空格分隔的值、其中一个值等于"val"的 E 元素

12. E[att|=val] 匹配所有 att 属性具有多个连字号分隔（hyphen-separated）的值、其中一个值以"val"开头的 E 元素，主要用于 lang 属性，比如"en"、"en-us"、"en-gb"等等

四、CSS 2.1 中的伪类

13. E:first-child 匹配父元素的第一个子元素

14. E:link 匹配所有未被点击的链接

15. E:visited 匹配所有已被点击的链接

16. E:active 匹配鼠标已经其上按下、还没有释放的 E 元素

17. E:hover 匹配鼠标悬停其上的 E 元素

18. E:focus 匹配获得当前焦点的 E 元素

19. E:lang(c) 匹配 lang 属性等于 c 的 E 元素

五、 CSS 2.1 中的伪元素

20. E:first-line 匹配 E 元素的第一行

21. E:first-letter 匹配 E 元素的第一个字母

22. E:before 在 E 元素之前插入生成的内容

23. E:after 在 E 元素之后插入生成的内容

六、CSS 3 的同级元素通用选择器

24. E ~ F 匹配任何在 E 元素之后的同级 F 元素

七、CSS 3 属性选择器

25. E[att^="val"] 属性 att 的值以"val"开头的元素

26. E[att$="val"] 属性 att 的值以"val"结尾的元素

27. E[att*="val"] 属性 att 的值包含"val"字符串的元素

八、CSS 3 中与用户界面有关的伪类

28. E:enabled 匹配表单中激活的元素

29. E:disabled匹配表单中禁用的元素

30. E:checked 匹配表单中被选中的 radio（单选框）或 checkbox（复选框）元素

31. E::selection 匹配用户当前选中的元素

九、CSS 3 中的结构性伪类

32. E:root 匹配文档的根元素，对于 HTML 文档，就是 HTML 元素

33. E:nth-child(n) 匹配其父元素的第 n 个子元素，第一个编号为 1

34. E:nth-last-child(n) 匹配其父元素的倒数第 n 个子元素，第一个编号为 1
35. E:nth-of-type(n) 与:nth-child()作用类似，但是仅匹配使用同种标签的元素
36. E:nth-last-of-type(n) 与:nth-last-child() 作用类似，但是仅匹配使用同种标签的元素
37. E:last-child 匹配父元素的最后一个子元素，等同于:nth-last-child(1)
38. E:first-of-type 匹配父元素下使用同种标签的第一个子元素，等同于:nth-of-type(1)
39. E:last-of-type 匹配父元素下使用同种标签的最后一个子元素，等同于:nth-last-of-type(1)
40. E:only-child 匹配父元素下仅有的一个子元素，等同于:first-child:last-child 或 :nth-child(1):nth-last-child(1)
41. E:only-of-type 匹配父元素下使用同种标签的唯一一个子元素，等同于:first-of-type:last-of-type 或 :nth-of-type(1):nth-last-of-type(1)
42. E:empty 匹配一个不包含任何子元素的元素，注意，文本节点也被看作子元素

十、CSS 3 的反选伪类

43. E:not(s) 匹配不符合当前选择器的任何元素

十一、CSS 3 中的 :target 伪类

44. E:target 匹配文档中特定"id"点击后的效果

26. 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？

设置 display 属性为 none，或者设置 visibility 属性为 hidden
设置宽高为 0，设置透明度为 0，设置 z-index 位置在-1000em
设置 text-indent:-9999px;

27. 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再具有 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）

28. 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{
        background-color:yellow;        /*ie8*/
        +background-color:pink;         /*ie7*/
        _background-color:orange;       /*ie6*/       }
```

更好的方式是使用 IE 条件判断语句：
<!-[if lte IE 6]>
内容
<![endif]->
等

29. 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)、边框(border)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)还有 border top/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。
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）
答案：<input> 、<img> 、<button> 、<texterea> 、<label>。

30. 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。

31. 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！）

32. 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。

33. 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
/*如何垂直居中一个<img>?（用更简便的方法。）*/
1   #container          /*<img>的容器设置如下*/
2   {
3           display:table-cell;
4           text-align:center;
5           vertical-align:middle;
6   }
```

34. px 和 em 的区别。

px 和 em 都是长度单位，区别是：
px 值固定，容易计算。
em 值不固定，是相对单位，其相对应父级元素的字体大小会调整

35. 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？

Reset 样式的目的是清除浏览器某些默认样式，方便 css 书写，例如：
* {margin:0;padding:0;list-style:none;}

Normalize 的理念与 reset 不同，它并不清除浏览器默认样式，而是尽量将所有浏览器的默认样式统一。

36. 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。

37. 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。

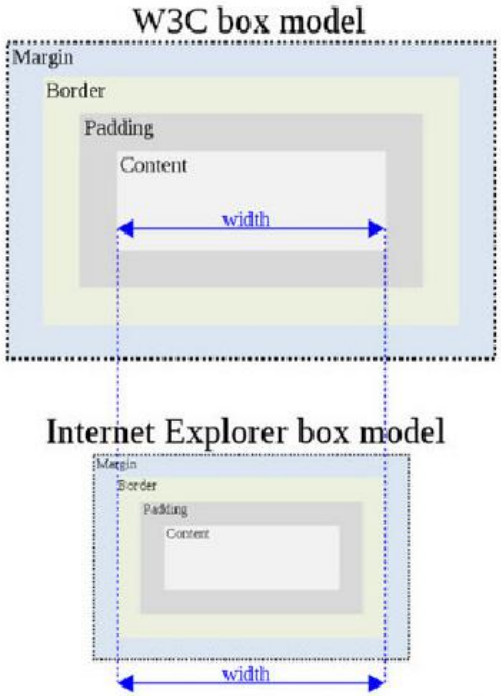
38. CSS 中 link 和@import 的区别是：

Link 属于 html 标签，而@import 是 CSS 中提供的
@import 有执行效率问题，它会打破浏览器并行加载资源，导致加载页面速度变慢。尽量不要使用@import。

39. 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。

40. 为什么要初始化样式？

用于浏览器默认 css 样式的存在并且不同浏览器对相同 HTML 标签的默认样式不同，若不初始化会造成不同浏览器之间的显示差异。

41. BFC 是什么？

BFC 就是“块级格式化上下文”的意思，创建了 BFC 的元素就是一个独立的盒子，不过只有 Block-level box 可以参与创建 BFC， 它规定了内部的 Block-level Box 如何布局，并且与这个独立盒子里的布局不受外部影响，当然它也不会影响到外面的元素。

BFC 有一下特性：

内部的 Box 会在垂直方向，从顶部开始一个接一个地放置。

Box 垂直方向的距离由 margin 决定。属于同一个 BFC 的两个相邻 Box 的 margin 会发生叠加

每个元素的 margin box 的左边， 与包含块 border box 的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

BFC 的区域不会与 float box 叠加。

BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之亦然。

计算 BFC 的高度时，浮动元素也参与计算。

如何触发 BFC？
float 除了 none 以外的值
overflow 除了 visible 以外的值（hidden，auto，scroll ）
display（table-cell，table-caption，inline-block，flex，inline-flex）
position 值为（absolute，fixed）
fieldset 元素

42. Doctype 的作用？严格模式与混杂模式的区别？

<!DOCTYPE>,文档声明；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

43. 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

44. 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

45. 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

46. 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。

47. 行内元素有哪些?块级元素有哪些?

答：块级元素：div p h1 h2 h3 h4 form ul

行内元素: a b br i span input select

48. 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。

49. 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。

50. 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>

51. 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)

52. 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。

53. 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 {}

54. 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。

55. 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。

56. 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.

57. css 的基本语句构成是?

选择器{属性 1:值 1;属性 2:值 2;……}

58. 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

59. 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

60. 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>

61. 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：
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。

55、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。

56、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

57、 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 ;

58、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;
```

59、 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。
```

60、 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）。
```

61、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
```

62、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
```

63、CSS 的盒模型由什么组成？

```
内容，border ,margin，padding
```

64、 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
```

65、 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 ;
```

66、 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、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？

低成本实现一套代码一个网页在多终端多设备下访问达到一定用户体验的开发方式。其布局会根据终端情况自适应调整达到一定水平的用户体验。

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. 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。

17. 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>
```

18. 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>
```

19. HTML5 引入什么新的表单属性？

```
Datalist  datetime  output  keygen  date  month  week  time  number  range
emailurl
```

20. 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 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。

21. 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。

22. 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；

23. 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]-->
```

24. 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。
```

25. 写出 HTML5 的文档声明方式

```
<DOCYPE  html>
```

26. 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...;
```

27. 自己对标签语义化的理解

```
网页使用什么 HTML 标签要看这个元素是什么元素，而不是看这个元素像什么元素。
例如我们用 h2 标签，是因为这个元素是二级标题 ，而不是因为它看起来比较字体比较粗比较大。
```

三、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
map()：遍历数组中的元素，返回一个新数组(包含回调函数返回的数据)
filter():遍历数组中的元素，返回一个新数组(包含回调函数返回 true 的元素)

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

功能一样，都是将当前函数作为指定对象的方法执行，即函数中的 this 是指定对象
call(thisObj，arg1,arg2...)　//将所有参数一个一个传递进去
apply(thisObj，[argArray])　//将所有参数放在数组中传入

9. Worker 继承 Person 的方法

```
//使用构造函数+原型的组合模式
function Person( age, name ){
   this.age = age;
   this.name = name;
}
Person.prototype.show = function(){
   alert('父级方法');
}

function Worker(age,name,job){
   Person.apply( this, arguments );
   this.job = job;
```

```
}
Worker.prototype = new Person();

var Person = new Person(14,'张三 ');
var Worker = new Worker (25,'李四','程序员');
```

10. 如何阻止事件冒泡和事件默认行为

```
//阻止事件冒泡
if(typeof ev.stopPropagation=='function') {   //标准的
        ev.stopPropagation();
} else { //非标准 IE
        window.event.cancelBubble = true;
}
//阻止事件默认行为
return false
```

11. 添加 删除 替换 插入到某个元素的方法

```
element.appendChild()
element.insertBefore()
element.replaceChild()
element.removeChild()
```

12. javascript 的内置对象和宿主对象

内置对象为 Object, Array, Function, Date, Math 等
宿主为浏览器自带的 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 原生中没有这个方法，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. 浏览器的同源策略

一段脚本(ajax)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、协议和端口号的组合

16. JavaScript 是一门什么样的语言，它有哪些特点？

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。
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, 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. 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
for (var i=0;i<len;i++) {
    if (domList[i].type ==  'checkbox' ) {
        checkBoxList.push(domList[i]);
    }
}
```

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. 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：<div onclick="test()"></div>
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
```

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
```

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
```

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);
```
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”。

```
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. 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. 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）

```
"<tr><td>{$id}</td><td>{$id}_{$name}</td></tr>".replace(/{\$id}/g, '10').replace(/{\$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. foo = foo||bar ，这行代码是什么意思？为什么要这样写？

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);
})()
```

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. 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. 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}。

```
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？

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);
};
```

答案：4 4 4。

原因：回调函数是在 for 结束之后才运行的。追问，如何让上述代码输出 1 2 3?

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

40. 写一个 function，清除字符串前后的空格。（兼容所有浏览器）

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

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

41. Javascript 中，以下哪条语句一定会产生运行错误？

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

42. 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'
```

43. var a=10; b=20; c=4;　++b+c+a++ 以下哪个结果是正确的？

```
A 34　 B、35　C、36　D、37　　答案(　B　)
```

44. 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=" ";
   }
}
```

45. 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”
```

46. 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 = //;
```

47. 以下哪个单词不属于 javascript 保留字：（B）

```
A.with
B.parent
C.class
D.void
```

48. 请选择结果为真的表达式：（C）

```
A.null instanceof Object
B.null === undefined
C.null == undefined
D.NaN == NaN
```

49. typeof 运算符返回值中有一个跟 javascript 数据类型不一致，它是_____Array_____。

50. 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_____ 。

51. 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

52. 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：7/21/2016

53. 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

54. 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

55. 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>
```

56. 截取字符串 abcdefg 的 efg

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

57. 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()
```

58. 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
```

59. 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");
```

60. 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
```

61. 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、 通过构造函数创建对象。
(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、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
```

62. 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
```

63. 请你谈谈 Cookie 的弊端？

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。

64. 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

65. 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

66. 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）

67. 判断一个字符串中出现次数最多的字符，统计这个次数

```
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');
```

68. 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;
        }
}
```

69. 事件委托是什么

```
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！
```

70. 闭包是什么，有什么特性，对页面有什么影响

```
答：当内部函数使用了外部函数的局部变量时，产生的一个对象(包含了所有使用了的变量)
                作用: 在函数执行完后，局部变量还会存在
        function outer(){
    var num = 1;
    function inner(){
        var n = 2;
        alert(n + num);
    }
```

```
    return inner;
}
var r = outer();
r();
```

71. 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
```

72. 字符串反转，如将 '12345678' 变成 '87654321'

```
//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() 转换为字符串
var str = '12345678';
str = str.split('').reverse().join('');
```

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

```
//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);
}
```

74. 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字！
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--;
        }
    }
}
```

75. 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;
}
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
```

76. 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());
```

77. window.location.search() 返回的是什么？

```
http://localhost:8080/xxx?ver=1.0&id=123
返回值：?ver=1.0&id=timlq 也就是问号后面的部分
```

78. window.location.reload() 作用？

```
    答：刷新当前页面。
```

79. javascript 中的垃圾回收机制？

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。

80. 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
```

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 。

81. 下面输出多少？

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

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;

82. 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); // “http://www.csser.com/”
```

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。）

83. a 输出多少？

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

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

84. 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
```

85. 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
```

86. 加减运算

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

87. 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
```

88. 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
```

89. 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
```

90. 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!");
```

91. 结果是：

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

双向非操作可以把字符串和数字转换为布尔值。

92. 声明对象，添加属性，输出属性

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

93. 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));
```

94. 检测变量类型

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


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

95. 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、绑定事件 IE： obj.attachEvent('onclick', method);
                FF: obj.addEventListener('click', method, false);
```

96. 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；
```

97. 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>
```

98. JS 中的简单继承 call 方法！

```
//定义个父母类，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();
```

99. 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。

100.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。
```

101.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
```

102.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
```

103.请描述出下列代码运行的结果

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

104.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>
```

105.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>
```

106. 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
```

107.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
```

108.、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

109. 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。

110.、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。。。

111.什么是三元表达式？“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。

112.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

113.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。

```
<div class=" warp" >
<div class=" left" ></div>
<div class=" right" ></div>
</div>
```

114.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);
}
```

115.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

116.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

117.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

118. 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

119.三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)
        }
    }
```

120.console.log(8 | 1); 输出值是多少？

```
答案：9
```

121.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。

```
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));
```

122.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。
```

123.一个 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]
```

124.、主流浏览器内核

```
IE trident          火狐 gecko          谷歌苹果 webkit          Opera：Presto
```

125.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);
        }
    }
```

126.JavaScript 的循环语句有哪些？

```
 for,for..in,while,do...while
```

127.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);
        };
    }
```

128.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”)

129.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}$
```

130. 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()

131.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。

132.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;
}
```

133.《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}$
```

134.《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());
```

135.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}$
```

136.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
}
```

137.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]));
```

138.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）;

139.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

140.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

141.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

142.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
");
            }
        }
}
```

143.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”

144.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
```

145.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
```

146.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]);
      }
}
```

147.程序中捕获异常的方法？

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

148.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')
```

149.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(' ');
};
```

150.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

151.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：

```
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

四、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

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：不能回退，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。

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

五、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、　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。

4、　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

5、　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

6、 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;
      }
   }
```

7、 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
```

8、 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();
```

9、 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);
}
```

10、 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;
    }
}
```

11、 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。

12、 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]) 。

13、 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);
  };
```

14、 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());
```

15、 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
            }
        });
    }
};
```

16、（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

给需要拖拽的节点绑定 mousedown, mousemove, mouseup 事件

mousedown 事件触发后，开始拖拽

mousemove 时，需要通过 event.clientX 和 clientY 获取拖拽位置，并实时更新位置

mouseup 时，拖拽结束

需要注意浏览器边界的情况

17、请实现如下功能


```
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();
})();
```

18、 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

19、 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。

20、 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。

21、 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）

22、 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

23、 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

24、 如何判断一个对象是否属于某个类？

instanceof

25、 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 。

26、 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

27、 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

28、 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

29、 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

30、 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

31、 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

32、 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

33、 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

34、 谈一谈你对 ECMAScript6 的了解？

ES6 新的语法糖，类，模块化等新特性

35、 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+')';
  }
}
```

36、 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

37、 DOM 操作——怎样添加、移除、移动、复制、创建和查找节点?

```
（1）创建新节点
     createDocumentFragment()    //创建一个 DOM 片段
     createElement()   //创建一个具体的元素
     createTextNode()   //创建一个文本节点
（2）添加、移除、替换、插入
     appendChild()
```

```
     removeChild()
     replaceChild()
     insertBefore()
（3）查找
     getElementsByTagName()    //通过标签名称
     getElementsByName()    //通过元素的 Name 属性的值
     getElementById()    //通过元素 Id，唯一性
```

38、 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
```

39、 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。
```

40、 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
```

41、 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

42、 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

43、 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

44、 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）

45、 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

46、 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

47、 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。

48、 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

49、 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！

50、 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。

51、 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_-]+)+$/;
```

52、 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

53、 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

54、 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。

55、 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=”utf-8”；

56、 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。

57、 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);
```

58、 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;
    }
};
```

59、 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');
}
```

60、 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);
```

61、 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'}
```

62、 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
```

63、 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
        };
    })();
```

64、 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
```

65、 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”。

66、 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

67、 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)
```

68、 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++) {}

69、 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。

70、 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

71、 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

六、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、 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

7、 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

8、 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

9、 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

10、 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）。

11、 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

12、 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。

七、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。）

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

八、NodeJs

1. 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。

2. 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）

3. 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）

4. (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

5. 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

6. 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）。

九、前端概括性问题

1. 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、AngularJs, ReactJs, 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：WebStorm, Hbuilder, 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）

2. 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)

3. 加班的看法

加班就像借钱，原则应当是------救急不救穷

4. 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

5. 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）

6. 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、react、angularjs、MVVM、MEAN

7. 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。

8. 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

9. 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。。。

10. 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

11. 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

12. 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

13. 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

14. 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

15. 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

16. 开发过程中遇到困难，如何解决。

考察解决问题的能力

17. 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？