1优雅降级和渐进增强的区别：

优雅降级：一开始就关注站点的完整功能，然后针对浏览器进行测试和修复。

优雅降级认为应该针对那些最高级的，最晚上的浏览器来设计网站，而那些被过时的有功能缺陷的浏览器下的测试工作安排在开发周期的最后阶段。

渐进增强：一开始值构建站点的最少特性，然后针对不同的追加浏览器。

应该关注内容本身

2：浏览器的兼容问题：

1：不同浏览器的默认标签的外补丁和内补丁不同。

解决方法：\*{margin:0;padding:0;}

2:块属性标签float后，又有横行的margin情况IE6情况下margin比设置的大：

解决方案：在float的标签样式中加入display:inline；将其转化成行内属性。

3：IE6识别是hacker是下划线\_和\*

IE7遨游是被的hacker是\*

3：如何对网页的文件和资源进行优化：

文件的合并：

减少调用其他页面，文件的数量；使用图片合并（雪碧图）

文件的压缩

将需要传输的文件内容压缩传到客户端子啊解压。将js,css文件进行压缩处理

使用CDN托管：

使用户就近下载所需的内容

缓存的应用：

4：前后端分离：

前端：负责view和control

后端：负责model层和业务逻辑的处理

5：关于浏览器的内核(渲染引擎)

浏览器的结构：用户的界面(UI) 包括菜单栏，工具栏，地址栏，前进和后退按钮，书签目录等，也就是能看到的除了页面的主窗口之外的部分

浏览器引擎：又称浏览器内核，渲染引擎，主要负责取得页面的内容，整理信息，计算页面的显示方式，然后会输出到显示器或者打印机：

Js解释器：也称js内核，主要负责处理javascript脚本程序，一般都会附带在浏览器之中，列如chrome的v8引擎；

网络部分：主要用于网络的调用，例如http请求，其接口和平台无关，并为所有的平台提供底层实现

数据存储：cookie，session。Localion

浏览器类型：

IE,Firefox,Safari,Chrome,Opera

内核分别是IE-trident

FF(mozilla) –gecko

Safari -- bink(webkit的分支)

Opera—bink()

浏览器加载文件

Reflow（重绘）：

在编译js文件的时候，浏览器会组织页面的渲染因此将js文件放到底部较好。

Js中有操作dom的地方。，浏览器会把页面重新渲染一边，这个过程叫做reflow

Css文件不会影响js文件的加载，但是会是影响js文件的执行，即使js文件内只有一行代码，也会造成阻塞；

常见能引起reflow的行为有：

改变窗口的大小：

改变文字的大小：

添加/删除样式表

脚本操作DOM

设置style属性

Reflow是不可避免的，只能经量减少，常用的方法有：

1：劲量不用行内style样式属性，操作元素样式的时候添加去掉class类的方法

2：给元素加动画的时候，可以把改元素定位设置成absolute,fixed,这样不会影响其他元素；

还有重绘repaint

进程和线程的区别：

一个程序至少有一个进程，一个进程至少有一个线程；

线程的划分尺度小于进程，使得多线程程序的并发性高。

线程是独立调度的基本单位，进程是拥有资源的基本单位

另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大提高了程序的运行效率；

线程在执行过程中与进程还是有区别的：

每个独立的线程有一个程序运行的入口，顺序执行序列和程序的出口

但是线程不能够独立的执行，必须依存在应用程序中，由应用程序提供多个线程执行控制

从逻辑的角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行，

前端优化问题：

减少http请求：css,data,url,spirit

Js，css的代码压缩；

前端模板JS+数据，减少由于HTML标签导致的带宽浪费，前端保存ajax请求，每次操作本地数据，不用请求，减少请求次数；

用innerHTMl代替dom操作，减少dom操作次数，优化js性能；

用setTimeOut避免页面失去相应；

缓存dom查找结果；

图片预载

避免使用table布局，table布局只有当数据加载完成，其内容也才会显示。显示比div+css慢

新增的css3：

新增的css3.0选择器（：not(input):所有不是class不是input的节点）

圆角border-radius

Text-shadow:box-shadow

旋转：transform

什么是无样式闪烁：

Wath:如果使用import方法对css进行导入，会导致某些页面在window下的internet explore出现一些奇怪的现象，以无样式页面的内容瞬间闪烁，这种现象称为文档样式暂时失效，简称FOUC

出现原因：使用import导入样式表

将样式表放在页面底部

有几个样式表，放到页面的不同位置

当样式表晚于机构性html加载，当加载到此样式表的时，页面停止之前的渲染。当样式表被加载和解析之后，将重新渲染页面，也就出现了短暂的花屏现象

解决方法：

使用link标签将样式表放到文档HEAD中，

* display:none  隐藏对应的元素，在文档布局中不再给它分配空间，它各边的元素会合拢，就当他从来不存在。
* visibility:hidden  隐藏对应的元素，但是在文档布局中仍保留原来的空间。
* 清除浮动是为了清除使用浮动元素产生的影响。浮动的元素，高度会塌陷，而高度的塌陷使我们页面后面的布局不能正常显示。
* **解释下浏览器是如何判断元素是否匹配某个 CSS 选择器？**
* 从后往前判断。  
  　　浏览器先产生一个元素集合，这个集合往往由最后一个部分的索引产生（如果没有索引就是所有元素的集合）。然后向上匹配，如果不符合上一个部分，就把元素从集合中删除，直到真个选择器都匹配完，还在集合中的元素就匹配这个选择器了。  
  　　举个例子，有选择器：
* body.ready #wrapper > .lol233
* 先把所有 class 中有 lol233 的元素拿出来组成一个集合，然后上一层，对每一个集合中的元素，如果元素的 parent id 不为 #wrapper 则把元素从集合中删去。 再向上，从这个元素的父元素开始向上找，没有找到一个 tagName 为 body 且 class 中有 ready 的元素，就把原来的元素从集合中删去。  
  　　至此这个选择器匹配结束，所有还在集合中的元素满足。

**CSS3的动画**

* 优点：
  + 1.在性能上会稍微好一些，浏览器会对CSS3的动画做一些优化（比如专门新建一个图层用来跑动画）
  + 2.代码相对简单
* 缺点：
  + 1.在动画控制上不够灵活
  + 2.兼容性不好
  + 3.部分动画功能无法实现（如滚动动画，视差滚动等）

**JavaScript的动画**

* 优点：
  + 1.控制能力很强，可以单帧的控制、变换
  + 2.兼容性好，写得好完全可以兼容IE6，且功能强大。
* 缺点：
  + 计算没有css快，另外经常需要依赖其他的库。

**结论**

* 所以，不复杂的动画完全可以用css实现，复杂一些的，或者需要交互的时候，用js会靠谱一些~

答: BFC指的是Block Formatting Context, 它提供了一个环境, html元素在这个环境中按照一定规则进行布局. 一个环境中的元素不会影响到其他环境中的布局. 决定了元素如何对其内容进行定位, 以及和其他元素的关系和相互作用.

　　其中: FC(Formatting Context): 指的是页面中的一个渲染区域, 并且拥有一套渲染规则, 它决定了其子元素如何定位, 以及与其他元素的相互关系和作用.

　　BFC: 块级格式化上下文, 指的是一个独立的块级渲染区域, 只有block-level box参与, 该区域拥有一套渲染规则来约束块级盒子的布局, 且与区域外部无关.

* BFC的生成:
  + 根元素
  + float的值不为none
  + overflow的值不为visible
  + display的值为 inline-block, table-cell, table-caption
  + position的值为absolute或fixed
* BFC的约束规则
  + 生成BFC元素的子元素会一个接一个的放置。垂直方向上他们的起点是一个包含块的顶部，两个相邻子元素之间的垂直距离取决于元素的margin特性。在BFC中相邻的块级元素外边距会折叠。
  + 生成BFC元素的子元素中，每一个子元素做外边距与包含块的左边界相接触，（对于从右到左的格式化，右外边距接触右边界），即使浮动元素也是如此（尽管子元素的内容区域会由于浮动而压缩），除非这个子元素也创建了一个新的BFC（如它自身也是一个浮动元素）。
  + 分解:
    1. 内部的Box会在垂直方向上一个接一个的放置
    2. 垂直方向上的距离由margin决定。（完整的说法是：属于同一个BFC的两个相邻Box的margin会发生重叠，与方向无关。）
    3. 每个元素的左外边距与包含块的左边界相接触（从左向右），即使浮动元素也是如此。（这说明BFC中子元素不会超出他的包含块，而position为absolute的元素可以超出他的包含块边界）
    4. BFC的区域不会与float的元素区域重叠
    5. 计算BFC的高度时，浮动子元素也参与计算
    6. 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元素，反之亦然
* BFC在布局中的应用
  + 不和浮动元素重叠:
    1. 如果一个浮动元素后面跟着一个非浮动元素, 就会产生覆盖
  + 防止margin重叠:
    1. 同一个BFC中的两个相邻Box才会发生重叠与方向无关，不过由于上文提到的第一条限制，我们甚少看到水平方向的margin重叠。这在IE中是个例外，IE可以设置write-mode
  + 解决浮动相关问题
    1. 父元素: overflow:hidden    IE: zoom:1(hasLayout)
    2. 根本原因在于创建BFC的元素，子浮动元素也会参与其高度计算，即不会产生高度塌陷问题。实际上只要让父元素生成BFC即可，并不只有这两种方式。
  + 多栏布局
    1. 比如左右两栏宽度固定, 中间栏自适应 则中间栏设置 overflow:hidden生成BFC
* H5的改进：

新元素：

画布canvasHTML5的canvas元素用于图形的绘制，通过脚本来完成

音频：audio

视频：vidio

语义性：article,nav，footer,section,asdio,hgroup;

时间time

新属性：

拖放：draggable <img draggable=”true”>

可编辑：contentedittable

新事件：

拖放： ondrag ondrop

关闭页面 onunload

窗口大小的改变：onresize

新的DOCTYPE声明 《!doctype html》

完全支持css3

Video 和 Audio

2d、3d绘制

本地存储webStorage

本地数据区SQL

Web应用

**2. 什么是语义化的html?**

答:

* what?
  + **根据内容的结构**（内容语义化），**选择合适的标签**（代码语义化）**便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析**。
* why?
  + 为了在没有CSS的情况下，页面也能呈现出很好地**内容结构、代码结构**:为了裸奔时好看；
  + **用户体验**：例如title、alt用于解释名词或解释图片信息、label标签的活用；
  + **有利于SEO**：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重
  + **方便其他设备解析**（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；
  + **便于团队开发和维护**，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。
* how?
  + 尽可能**少的使用无语义的标签**div和span；
  + 在语义不明显时，既可以使用div或者p时，**尽量用p, 因为p在默认情况下有上下间距，**对兼容特殊终端有利；
  + **不要使用纯样式标签**，如：b、font、u等，改用css设置。
  + 需要强调的文本，可以包含在strong或者em标签中（浏览器预设样式，能用CSS指定就不用他们），**strong默认样式是加粗（不要用b），em是斜体（不用i）**；
  + 使用**表格**时，标题要用caption，表头用thead，主体部分用tbody包围，尾部用tfoot包围。表头和一般单元格要区分开，表头用th，单元格用td；
  + **表单域**要用fieldset标签包起来，并用legend标签说明表单的用途；
  + 每个**input标签**对应的说明文本都需要使用**label标签**，并且通过为input设置id属性，在lable标签中设置**for=someld**来让说明文本和相对应的input关联起来。

　　参考自: [理解HTML语义化](http://www.cnblogs.com/freeyiyi1993/p/3615179.html)

**3. 从前端角度出发谈谈做好seo需要考虑什么?**

答:

* 语义化html标签
* 合理的title, description, keywords;
* 重要的html代码放前面
* 少用iframe, 搜索引擎不会抓取iframe中的内容
* 图片加上alt

**4. 文档类型(DOCTYPE)**

答:

* **作用:** doctype声明位于文档中最前面的位置，处于标签之前，告知浏览器使用的是哪种规范。
* **类型:** 三种: Strict、Transitional 以及 Frameset
* **如果不声明:**不写doctype，浏览器会进入quirks mode （混杂模式）。即，如果不声明doctype，浏览器不引入w3c的标准，那么早期的浏览器会按照自己的解析方式渲染页面。浏览器采用自身方式解析页面的行为称为"quirks mode（混杂模式也称怪异模式）"；采用w3c方式解析就是"strict mode（标准模式）"。 如果完全采用strictmode就不会出任何的差错，但这样会降低程序的容错率，加重开发人员的难度
* **用哪种:**
  + 没有doctype声明的采用quirks mode解析
  + 对于有doctype的大多数采用standard mord。
  + 特殊情况：
    - 对于那些浏览器不能识别的doctype ,浏览器采用quirks mode；
    - 没有声明DTD或者html版本声明低于4.0采用quirks mode，其他使用standard mode；
    - ie6中，如果在doctype声明前有一个xml声明(比如:<?xml version="1.0" encoding="iso-8859-1"?>)，则采用quirks mode解析
* **标准模式与怪异模式的区别:**
  + 标准模式：浏览器根据规范呈现页面
  + 混杂模式（怪异模式）：页面以一种*比较宽松的兼容方式*显示。
  + 他们最大的不同是对盒模型的解析。
    - 在strict mode中 ：width是内容宽度 ，也就是说,元素真正的宽度 = margin-left + border-left-width + padding-left + width + padding-right + border-right- width +  margin-right;
    - 在quirks mode中 ：width则是元素的实际宽度 ，内容宽度 = width - (margin-left + margin-right + padding-left + padding-right + border-left-width + 　border-right-width)

**5. 使用XHTML的局限有哪些?**

　　答：

* XHTML较为严格，标签必须闭合，必须要body，head等
* 如果页面使用 'application/xhtml+xml' 一些老的浏览器并不兼容

**6. 如果网页内容需要多语言,要怎么做?**

　　答: 采用统一编码utf-8模式

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <META HTTP-EQUIV=“Content-Type” CONTENT=“text/html; CHARSET=字符集">  不写，根据浏览器默认字符集显示  charset=gb2312 简体中文  charset=big5 繁体中文  charset=EUC\_KR 韩语  charset=Shift\_JIS 或 EUC\_JP 日语  charset= KOI8-R / Windows-1251 俄语  charset=iso-8859-1 西欧语系（荷兰语,英语,法语,德语,意大利语,挪威语,葡萄牙语,瑞士语.等十八种语言）  charset=iso-8859-2 中欧语系  charset=iso-8859-5 斯拉夫语系（保加利亚语,Byelorussian语,马其顿语,俄语,塞尔维亚语,乌克兰语等）  charset=uft-8 unicode多语言 |

　　参考自: [多语言的网站怎么做呀](http://www.cnblogs.com/luluping/archive/2009/04/10/1433108.html) [多语言网站设计需要注意的问题](http://handyxuefeng.blog.163.com/blog/static/454521722013111732534620/)

**7. data-\*属性的作用**

* html5规范里增加了一个自定义data属性
* 为前端开发者提供自定义的属性，这些属性集可以通过对象的dataset属性获取，不支持该属性的浏览器可以通过 getAttribute方法获取  
  <div data-author="david" data-time="2011-06-20" data-comment-num="10">...</div>  
  div.dataset.commentNum; // 可通过js获取 10
* data-为前端开发者提供自定义的属性，这些属性集可以通过对象的dataset属性获取，不支持该属性的浏览器可以通过 getAttribute方法获取。ppk提到过使用rel属性，lightbox库推广了rel属性，HTML5提供了data-做替代，这样可以更好 地使用自定义的属性
* 需要注意的是，data-之后的以连字符分割的多个单词组成的属性，获取的时候使用驼峰风格。

**8. 如果把 HTML5 看作做一个开放平台，那它的构建模块有哪些？**

答:

　　　　1）Web Storage API  
　　　　2）基于位置服务LBS  
　　　　3）无插件播放音频视频  
　　　　4）调用相机和GPU图像处理单元等硬件设备  
　　　　5）拖拽和Form API

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

答:

* **cookie:**
  + cookie是网站为了**标示用户身份**而储存在用户本地终端（Client Side）上的数据（通常经过加密）。
  + cookie数据**始终在同源的http请求中携带**（即使不需要），记会在浏览器和服务器间来回传递。
* **sessionStorage和localStorage**不会自动把数据发给服务器，仅在本地保存。
* **存储大小：**
  + cookie数据大小不能超过4k。
  + sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。
* **有期时间**：
  + localStorage    存储**持久数据**，浏览器关闭后数据不丢失除非主动删除数据；
  + sessionStorage  数据在当前浏览器窗口关闭后自动删除。
  + cookie          设置的cookie**过期时间之前**一直有效，即使窗口或浏览器关闭
* **作用域不同:**
  + sessionStorage**不在**不同的浏览器窗口中共享，即使是同一个页面；
  + localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。
* Web Storage 支持**事件通知机制**，可以将数据更新的通知发送给监听者。
* Web Storage 的 **api 接口**使用更方便。

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

sessionStorage 和 localStorage 是HTML5 Web Storage API 提供的，这两种方式都允许开发者使用js设置的键值对进行操作，在在重新加载不同的页面的时候读出它们。这一点与cookie类似。可以方便的在web请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。

sessionStorage、localStorage、cookie都是在浏览器端存储的数据，

其中sessionStorage的概念很特别，引入了一个“浏览器窗口”的概念。 sessionStorage是在同源的同窗口（或tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，sessionStorage即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage对象也是不同的。

Web Storage 数据完全存储在客户端, 不需要通过浏览器的请求将数据传给服务器, 因此比cookies能够存储更多的数据,大概5M左右

Web Storage带来的好处：使用 local storage和session storage主要通过在js中操作这两个对象来实现，分别为window.localStorage和window.sessionStorage. 这两个对象均是Storage类的两个实例，自然也具有Storage类的属性和方法。

减少网络流量：一旦数据保存在本地后，就可以避免再向服务器请求数据，因此减少不必要的数据请求，减少数据在浏览器和服务器间不必要地来回传递。

快速显示数据：性能好，从本地读数据比通过网络从服务器获得数据快得多，本地数据可以即时获得。再加上网页本身也可以有缓存，因此整个页面和数据都在本地的话，可以立即显示。

临时存储：很多时候数据只需要在用户浏览一组页面期间使用，关闭窗口后数据就可以丢弃了，这种情况使用sessionStorage非常方便。

[复制代码](javascript:void(0);)

**10. 浏览器本地存储与服务器端存储之间的区别**

* 其实数据既可以在浏览器本地存储，也可以在服务器端存储。
* 浏览器端可以保存一些数据，需要的时候直接从本地获取，sessionStorage、localStorage和cookie都由浏览器存储在本地的数据。
* 服务器端也可以保存所有用户的所有数据，但需要的时候浏览器要向服务器请求数据。
  + 1.服务器端可以保存用户的**持久数据**，如数据库和云存储将用户的大量数据保存在服务器端。
  + 2.服务器端也可以保存用户的**临时会话数据**。服务器端的session机制，如jsp的 session 对象，数据保存在服务器上。实现上，服务器和浏览器之间仅需传递session id即可，服务器根据session id找到对应用户的session对象。会话数据仅在一段时间内有效，这个时间就是server端设置的session有效期。
* 服务器端保存所有的用户的数据，所以服务器端的开销较大，而浏览器端保存则把不同用户需要的数据分布保存在用户各自的浏览器中。
* 浏览器端一般只用来存储小数据，而服务器可以存储大数据或小数据。
* 服务器存储数据安全一些，浏览器只适合存储一般数据。

**11. sessionStorage和页面js数据对象的区别**

答:

* 页面中一般的 js 对象或数据的生存期是仅在当前页面有效，因此刷新页面或转到另一页面这样的重新加载页面的情况，数据就不存在了。
* 而sessionStorage 只要同源的同窗口（或tab）中，刷新页面或进入同源的不同页面，数据始终存在。也就是说只要这个浏览器窗口没有关闭，加载新页面或重新加载，数据仍然存在。

**12. canvas和svg的区别?**

答:

* **svg:**
  + SVG是一种使用XML描述2D图形的语言
  + SVG基于XML, 这意味着SVG DOM中的每个元素都是可用的. 所以可以为每个元素附加JavaScript事件处理器
  + 在SVG中, 每个被绘制的图像均被视为对象. 如果SVG对象的属性发生变化, 那么浏览器能够自动重现图像
* **Canvas**
  + Canvas通过js来绘制2D图形
  + Canvas是逐像素进行渲染的
  + 在Canvas中, 一旦图形被绘制完成, 它就不会继续得到浏览器的关注. 如果其位置发生变化, 那么整个场景也需要重新绘制, 包括任何或许已被图形覆盖的对象.
* **区别**
  + Canvas支持分辨率, SVG不支持
  + Canvas不支持事件处理器, SVG支持
  + Canvas只有弱的文本渲染能力, 而SVG最适合带有大型渲染区域的应用程序(比如谷歌地图)
  + Canvas能够以.png或.jpg格式保存结果图像
  + SVG的复杂度过高的话会减慢渲染速度(任何过度使用DOM的应用都不快)
  + Canvas最适合图像密集型的游戏, 其中的许多对象会被频繁重绘. 而SVG不适合游戏应用
  + Canvas是基于位图的图像,它不能够改变大小, 只能缩放显示; SVG是基于矢量的, 所以它能够很好地处理图形大小的改变
  + Canvas提供的功能更原始, 适合**像素处理, 动态渲染和大数据量绘制**; SVG功能更完善, 适合**静态图片显示, 高保真文档查看和打印**的应用场景
  + 绘制Canvas对象后, 不能使用脚本和CSS对它进行修改; 而SVG对象是文档对象模型的一部分, 所以可以随时使用脚本和CSS修改它们

**13. src和href的区别?**

答:

* src指向外部资源的位置, 用于替换当前元素, 比如js脚本, 图片等元素
* href指向网络资源所在的位置, 用于在当前文档和引用资源间确定联系, 加载css

**四、js**

**1. ajax, 跨域, jsonp**

　　参考： [《JavaScript》高级程序设计第21章：Ajax和Comet](http://www.cnblogs.com/haoyijing/p/5778155.html)

[js跨域](http://www.cnblogs.com/haoyijing/p/5760423.html#z6)

[jQuery中Ajax操作](http://blog.csdn.net/u013194113/article/details/52034106#t1)

**2. apply和call的用法和区别:**

* 用法: 都能继承另一个对象的方法和属性,区别在于参数列表不一样
* 区别:
  + Function.apply(obj, args) args是一个数组,作为参数传给Function
  + Function.call(obj, arg1, arg2,...)  arg\*是参数列表
  + apply一个妙用: 可以将一个数组默认的转化为一个参数列表
    - 举个栗子: 有一个数组arr要push进一个新的数组中去, 如果用call的话需要把数组中的元素一个个取出来再push, 而用apply只有Array.prototype.push.apply(this, arr)

　　参考自: [区别和详解：js中call()和apply()的用法](http://www.cnblogs.com/wdlhao/p/5614522.html)

**3. bind函数的兼容性**

* 用法:
  + bind()函数会创建一个新函数, 为绑定函数。当调用这个绑定函数时,绑定函数会以创建它时传入bind方法的第一个参数作为this,传入bind方法的第二个以及以后的参数加上绑定函数运行时本身的参数按照顺序作为原函数的参数来调用原函数.
  + 一个绑定函数也能使用[new](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/new)操作符创建对象：这种行为就像把原函数当成构造器。提供的 this 值被忽略，同时调用时的参数被提供给模拟函数。

**4. 解释下事件代理**

答: 事件委托利用了事件冒泡, 只指定一个事件处理程序, 就可以管理某一类型的所有事件.

例:

html部分: 要点击li弹出其id

[复制代码](javascript:void(0);)

<ul id="list">

<li id="li-1">Li 2</li>

<li id="li-2">Li 3</li>

<li id="li-3">Li 4</li>

<li id="li-4">Li 5</li>

<li id="li-5">Li 6</li>

<li id="li-6">Li 7</li>

</ul>

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

//js部分  
document.getElementById("list").addHandler("click", function(e){

var e = e || window.event;

var target = e.target || e.srcElement;

if(target.nodeName.toUpperCase == "LI"){

console.log("List item", e,target.id, "was clicked!");

}

});

[复制代码](javascript:void(0);)

**5. 解释下js中this是怎么工作的?**

答:

　　this 在 JavaScript 中主要由以下五种使用场景。

* 作为函数调用，this 绑定全局对象，浏览器环境全局对象为 window 。
* 内部函数内部函数的 this 也绑定全局对象，应该绑定到其外层函数对应的对象上，这是 JavaScript的缺陷，用that替换。
* 作为构造函数使用，this 绑定到新创建的对象。
* 作为对象方法使用，this 绑定到该对象。
* 使用apply或call调用 this 将会被显式设置为函数调用的第一个参数。

**6. 继承**

　　参考：[js怎么实现继承？](http://www.cnblogs.com/haoyijing/p/5760586.htm)

**7. AMD vs. CommonJS？**

答: AMD是依赖提前加载,CMD是依赖延时加载

**8. 什么是哈希表?**

答:  哈希表（Hash table，也叫散列表），是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数，存放记录的数组叫做散列表。

　　使用哈希查找有两个步骤:

1. 使用哈希函数将被查找的键转换为数组的索引。在理想的情况下，不同的键会被转换为不同的索引值，但是在有些情况下我们需要处理多个键被哈希到同一个索引值的情况。所以哈希查找的第二个步骤就是处理冲突
2. 处理哈希碰撞冲突。有很多处理哈希碰撞冲突的方法，比如拉链法和线性探测法。

元素特征转变为数组下标的方法就是散列法。散列法当然不止一种，下面列出三种比较常用的：

**1，除法散列法**最直观的一种，上图使用的就是这种散列法，公式：   
      index = value % 16   
学过汇编的都知道，求模数其实是通过一个除法运算得到的，所以叫“除法散列法”。

**2，平方散列法**求index是非常频繁的操作，而乘法的运算要比除法来得省时（对现在的CPU来说，估计我们感觉不出来），所以我们考虑把除法换成乘法和一个位移操作。公式：   
      index = (value \* value) >> 28   **（**右移，除以2^28。记法：左移变大，是乘。右移变小，是除。**）**如果数值分配比较均匀的话这种方法能得到不错的结果，但我上面画的那个图的各个元素的值算出来的index都是0——非常失败。也许你还有个问题，value如果很大，value \* value不会溢出吗？答案是会的，但我们这个乘法不关心溢出，因为我们根本不是为了获取相乘结果，而是为了获取index。

**3，斐波那契（Fibonacci）散列法**

解决冲突的方法:

**1. 拉链法**

　　将大小为M 的数组的每一个元素指向一个条链表，链表中的每一个节点都存储散列值为该索引的键值对，这就是拉链法.

　　对采用拉链法的哈希实现的查找分为两步，首先是根据散列值找到等一应的链表，然后沿着链表顺序找到相应的键。

**2. 线性探测法:**

　　使用数组中的空位解决碰撞冲突

参考: [哈希表的工作原理](http://www.cnblogs.com/ghj1976/p/3689917.html)  　[浅谈算法和数据结构: 十一 哈希表](http://www.cnblogs.com/yangecnu/p/Introduce-Hashtable.html)

**9. 什么是闭包? 闭包有什么作用?**

答:

* 闭包是指有权访问另一个函数作用域中的变量的函数. 创建闭包常见方式,就是在一个函数内部创建另一个函数.
* 作用:
  + 匿名自执行函数  (function (){ ... })();   创建了一个匿名的函数，并立即执行它，由于外部无法引用它内部的变量，因此在执行完后很快就会被释放，关键是这种机制不会污染全局对象。
  + 缓存, 可保留函数内部的值
  + 实现封装
  + 实现模板

　　参考自: [js闭包的用途](http://blog.csdn.net/sunlylorn/article/details/6534610)

**10. 伪数组:**

* 什么是伪数组:
  + 伪数组是能通过Array.prototype.slice **转换为真正的数组**的**带有length属性**的对象
  + 比如arguments对象,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
  + 我们可以通过Array.prototype.slice.call(fakeArray)将伪数组转变为真正的Array对象: 返回新数组而不会修改原数组

　　参考: [伪数组](http://www.cnblogs.com/fool/archive/2010/10/09/1846966.html)

**11. undefined和null的区别, 还有undeclared:**

* null表示没有对象, 即此处不该有此值. 典型用法:
  + （1） 作为函数的参数，表示该函数的参数不是对象。
  + （2） 作为对象原型链的终点。
  + ( 3 )   null可以作为空指针. 只要意在保存对象的值还没有真正保存对象,就应该明确地让该对象保存null值.
* undefined表示缺少值, 即此处应该有值, 但还未定义.
  + （1）变量被声明了，但没有赋值时，就等于undefined。

（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

* undeclared即为被污染的命名, 访问没有被声明的变量, 则会抛出异常, 终止执行. 即undeclared是一种语法错误

　　参考: [undefined与null的区别](http://www.ruanyifeng.com/blog/2014/03/undefined-vs-null.html)

**12. 事件冒泡机制:**

从目标元素开始，往顶层元素传播。途中如果有节点绑定了相应的事件处理函数，这些函数都会被一次触发。如果想阻止事件起泡，可以使用e.stopPropagation()（Firefox）或者e.cancelBubble=true（IE）来组织事件的冒泡传播。

**13. 解释下为什么接下来这段代码不是 IIFE(立即调用的函数表达式)：function foo(){ }();?**

答:

　　而**函数定义（语句以function关键字开始）是不能被立即执行**的，这无疑会导致语法的错误（SyntaxError）。当函数定义代码段包裹在括号内，使解析器可以将之识别为函数表达式，然后调用。IIFE:  *(function foo(){})()*

　　区分  (function(){})(); 和  (function(){}());  其实两者实现效果一样。  
　　函数字面量：首先声明一个函数对象，然后执行它。(function () { alert(1); })();

　　优先表达式：由于Javascript执行表达式是**从圆括号里面到外面，所以可以用圆括号强制执行声明的函数**。(function () { alert(2); }());

**14. "attribute" 和 "property" 的区别是什么？**

答:

　　DOM元素的attribute和property两者是不同的东西。attribute翻译为“特性”，property翻译为“属性”。  
　　attribute是一个特性节点，每个DOM元素都有一个对应的attributes属性来存放所有的attribute节点，attributes是一个类数组的容器，说得准确点就是NameNodeMap，不继承于Array.prototype，不能直接调用Array的方法。attributes的每个数字索引以名值对(name=”value”)的形式存放了一个attribute节点。<div class="box" id="box" gameid="880">hello</div>

　　property就是一个属性，如果把DOM元素看成是一个普通的Object对象，那么property就是一个以名值对(name=”value”)的形式存放在Object中的属性。要添加和删除property和普通的对象类似。

　　很多attribute节点还有一个相对应的property属性，比如上面的div元素的id和class既是attribute，也有对应的property，不管使用哪种方法都可以访问和修改。

　　总之，attribute节点都是在HTML代码中可见的，而property只是一个普通的名值对属性。

**15. 请指出 document load 和 document ready 两个事件的区别。**

* document.ready和onload的区别——JavaScript文档加载完成事件。页面加载完成有两种事件:
  + 一是ready，表示文档结构已经加载完成（不包含图片等非文字媒体文件）
  + 二是onload，指示页面包含图片等文件在内的所有元素都加载完成。
* jQuery中$(function(){/\* do something\*/});他的作用或者意义就是:在DOM加载完成后就可以可以对DOM进行操作。一般情况先一个页面响应加载的顺序是，域名解析-加载html-加载js和css-加载图片等其他信息。

**16. 什么是use strict? 其好处坏处分别是什么?**

答:

　　在所有的函数 (或者所有最外层函数) 的开始处加入 "use strict"; 指令启动严格模式。

　　"严格模式"有两种调用方法

　　　　　　1）将"use strict"放在脚本文件的第一行，则整个脚本都将以"严格模式"运行。如果这行语句不在第一行，则无效，整个脚本以"正常模式"运行。如果不同模式的代码文件合并成一个文件，这一点需要特别注意。

　　　　　　2）将整个脚本文件放在一个立即执行的匿名函数之中。

**好处**

　　　　　　- 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

　　　　　　- 消除代码运行的一些不安全之处，保证代码运行的安全；

　　　　　　- 提高编译器效率，增加运行速度；

　　　　　　- 为未来新版本的Javascript做好铺垫。

**坏处**

　　　　　　同样的代码，在"严格模式"中，可能会有不一样的运行结果；一些在"正常模式"下可以运行的语句，在"严格模式"下将不能运行

**17. 浏览器端的js包括哪几个部分?**

　　答: 核心( ECMAScript) , 文档对象模型(DOM), 浏览器对象模型(BOM)

**18. DOM包括哪些对象?**

　　答: DOM是针对HTML和XML文档的一个API(应用程序编程接口). DOM描绘了一个层次化的节点树, 允许开发人员添加, 移除和修改页面的某一部分.

* 常用的DOM方法:
  + getElementById(id)
  + getElementsByTagName()
  + appendChild(node)
  + removeChild(node)
  + replaceChild()
  + insertChild()
  + createElement()
  + createTextNode()
  + getAttribute()
  + setAttribute()
* 常用的DOM属性
  + innerHTML  节点(元素)的文本值
  + parentNode  节点(元素)的父节点
  + childNodes
  + attributes   节点(元素)的属性节点

　　参考:  [HTML DOM 方法](http://www.w3school.com.cn/htmldom/dom_methods.asp)

**19. js有哪些基本类型?**

　　答: Undefined, Null, Boolean, Number, String

　　　　Object是复杂数据类型, 其本质是由一组无序的名值对组成的.

**20. 基本类型与引用类型有什么区别?**

　　答:

* 基本类型如上题所示. 引用类型则有: Object, Array, Date, RegExp, Function
* 存储
  + 基本类型值在内存中占据固定大小的空间,因此被保存在栈内存中
  + 引用类型的值是对象, 保存在堆内存中. 包含引用类型的变量实际上包含的并不是对象本身, 而是一个指向改对象的指针
* 复制
  + 从一个变量向另一个变量复制基本类型的值, 会创建这个值的一个副本
  + 从一个变量向另一个变量复制引用类型的值, 复制的其实是指针,　因此两个变量最终都指向同一个对象
* 检测类型
  + 确定一个值是哪种基本类型可以用typeof操作符,
  + 而确定一个值是哪种引用类型可以使用instanceof操作符

　　参考: 《JavaScript高级程序设计》

**21. 关于js的垃圾收集例程**

　　js是一门具有自动垃圾回收机制的编程语言,开发人员不必关心内存分配和回收问题

* 离开作用域的值将被自动标记为可以回收, 因此将在垃圾收集期间被删除
* "标记清除"是目前主流的垃圾收集算法, 这种算法的思路是给当前不使用的值加上标记, 然后再回收其内存
* 另一种垃圾收集算法是"引用计数", 这种算法的思想是跟踪记录所有值被引用的次数. js引擎目前都不再使用这种算法, 但在IE中访问非原生JS对象(如DOM元素)时, 这种算法仍然可能会导致问题
* 当代码中存在循环引用现象时, "引用计数" 算法就会导致问题
* 解除变量的引用不仅有助于消除循环引用现象, 而且对垃圾收集也有好处. 为了确保有效地回收内存, 应该及时解除不再使用的全局对象, 全局对象属性以及循环引用变量的引用

**22. ES5中, 除了函数,什么能够产生作用域?**

答: try-catch 和with延长作用域. 因为他们都会创建一个新的变量对象.

　　这两个语句都会在作用域链的前端添加一个变量对象. 对with语句来说, 会将指定的对象添加到作用域链中. 对catch语句来说, 会创建一个新的变量对象, 其中包含的是被抛出的错误对象的声明.

* 当try代码块中发生错误时，执行过程会跳转到catch语句，然后把异常对象推入一个可变对象并置于作用域的头部。在catch代码块内部，函数的所有局部变量将会被放在第二个作用域链对象中。请注意，一旦catch语句执行完毕，作用域链机会返回到之前的状态。try-catch语句在代码调试和异常处理中非常有用，因此不建议完全避免。你可以通过优化代码来减少catch语句对性能的影响。一个很好的模式是将错误委托给一个函数处理
* with(object) {statement}。它的意思是把object添加到作用域链的顶端
  + e.g.
  + [复制代码](javascript:void(0);)
  + function buildUrl(){
  + var qs = "?debug=true";
  + //with接收location对象, 因此其变量对象中就包含了location对象的所有属性和方法, 而这个变量对象被添加到了作用域链的前端
  + with(location){
  + //这里的href其实是location.href. 创建了一个名为url的变量, 就成了函数执行环境的一部分
  + var url = href + qs;
  + }
  + return url;

}

[复制代码](javascript:void(0);)

　　参考: [js try、catch、finally语句还有with语句](http://blog.csdn.net/dys1990/article/details/6857585)  [JavaScript 开发进阶：理解 JavaScript 作用域和作用域链](http://www.cnblogs.com/lhb25/archive/2011/09/06/javascript-scope-chain.html)

**23. js有几种函数调用方式?**

答:

* 方法调用模型    var obj = { func : function(){};}    obj.func()
* 函数调用模式　　var func = function(){}    func();
* 构造器调用模式
* apply/ call调用模式

**24. 描述事件模型?IE的事件模型是怎样的？事件代理是什么？事件代理中怎么定位实际事件产生的目标？**

答: 　捕获->处于目标->冒泡，IE应该是只有冒泡没有捕获。  
　　  事件代理就是在父元素上绑定事件来处理，通过event对象的target来定位。

**25. js动画有哪些实现方法?**

答: 用定时器 setTimeout和setInterval

**26. 还有什么实现动画的方法?**

答:

* js动画: 使用定时器
* CSS : transition , animation
  + transition 包含4种属性：transition-delaytransition-durationtransition-propertytransition-timing-function，对应动画的4种属性： 延迟、持续时间、对应css属性和缓动函数，
  + transform 包含7种属性：animation-nameanimation-durationanimation-timing-functionanimation-delayanimation-directionanimation-iteration-countanimation-fill-modeanimation-play-state，它们可以定义动画名称，持续时间，缓动函数，动画延迟，动画方向，重复次数，填充模式。
* HTML5 动画
  + canvas
  + svg
  + webgl

　　参考自: [前端动画效果实现的简单比较](https://segmentfault.com/a/1190000000594926)

**27. 面向对象有哪几个特点?**

答: 封装, 继承, 多态

**28. 如何判断属性来自自身对象还是原型链?**

答: hasOwnPrototype

**29. ES6新特性**

答:

　　1)**箭头操作符** inputs=>outputs: 操作符左边是输入的参数，而右边则是进行的操作以及返回的值

　　2) **支持类**, 引入了**class**关键字. ES6提供的类实际上就是JS原型模式的包装

　　3) **增强的对象字面量**.

　　　　1. 可以在对象字面量中**定义原型  \_\_proto\_\_**: xxx  //设置其原型为xxx,相当于继承xxx

　　　　2. 定义方法可以**不用function**关键字

　　　　3. **直接调用父类方法**

　　4) **字符串模板**: ES6中允许使用反引号 ` 来创建字符串，此种方法创建的字符串里面可以包含由美元符号加花括号包裹的变量${vraible}。

　　5) **自动解析数组或对象中的值**。比如若一个函数要返回多个值，常规的做法是返回一个对象，将每个值做为这个对象的属性返回。但在ES6中，利用解构这一特性，可以直接返回一个数组，然后数组中的值会自动被解析到对应接收该值的变量中。

　　6) **默认参数值:** 现在可以在定义函数的时候指定参数的默认值了，而不用像以前那样通过逻辑或操作符来达到目的了。

　　7) **不定参数**是在函数中使用命名参数同时接收不定数量的未命名参数。在以前的JavaScript代码中我们可以通过arguments变量来达到这一目的。不定参数的格式是三个句点后跟代表所有不定参数的变量名。比如下面这个例子中，…x代表了所有传入add函数的参数。

　　8) **拓展参数**则是另一种形式的语法糖，它允许传递数组或者类数组直接做为函数的参数而不用通过apply。

　　9) **let和const关键字:** 可以把let看成var，只是它定义的变量被限定在了特定范围内才能使用，而离开这个范围则无效。const则很直观，用来定义常量，即无法被更改值的变量。

　　10)**for of值遍历**每次循环它提供的不是序号而是值。

　　11) **iterator, generator**

　　12) **模块**

　　13) **Map, Set, WeakMap, WeakSet**

　　14) **Proxy**可以监听对象身上发生了什么事情，并在这些事情发生后执行一些相应的操作。一下子让我们对一个对象有了很强的追踪能力，同时在数据绑定方面也很有用处。

　　15)**Symbols**Symbol 通过调用symbol函数产生，它接收一个可选的名字参数，该函数返回的symbol是唯一的。之后就可以用这个返回值做为对象的键了。Symbol还可以用来创建私有属性，外部无法直接访问由symbol做为键的属性值。

　　16) **Math, Number, String, Object的新API**

17) **Promises**是处理异步操作的一种模式

　　参考: [ES6新特性概览](http://www.cnblogs.com/Wayou/p/es6_new_features.html)

**30. 如何获取某个DOM节点，节点遍历方式**

答: getElementById()  getElementsByTagName()

节点遍历: [先序遍历DOM树的5种方法](http://www.cnblogs.com/tracylin/p/5220867.html)

**31. 用LESS如何给某些属性加浏览器前缀？**

答:  可以自定义一个函数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | .border-radius(@values) {    -webkit-border-radius: @values;       -moz-border-radius: @values;            border-radius: @values;  }  div {    .border-radius(10px);  } |

**32.  js异步模式如何实现？**

答: [JavaScript异步编程的Promise模式](http://www.cnblogs.com/haoyijing/p/5737951.html)

**33. 事件机制，如何绑定事件处理函数。**

**34. 图片预加载的实现**

答:

1. [使用jQuery图片预加载插件Lazy Load](http://www.cnblogs.com/haoyijing/p/5818236.html#imgLoad1)
   1. 加载jQuery, 与jquery.lazyload.js
   2. 设置图片的占位符为data-original, 给图片一个特别的标签,比如class=".lazy"
   3. 然后延迟加载: $('img.lazy').lazyload();这个函数可以选择一些参数:
      1. 图片预先加载距离：threshold，通过设置这个值，在图片未出现在可视区域的顶部距离这个值时加载。
      2. 事件绑定加载的方式：event
      3. 图片限定在某个容器内：container
2. [使用js实现图片加载](http://www.cnblogs.com/haoyijing/p/5818236.html#imgLoad2): 就是new一个图片对象, 绑定onload函数, 赋值url
3. [用CSS实现图片的预加载](http://www.cnblogs.com/haoyijing/p/5818236.html#imgLoad3)
   1. 写一个CSS样式设置一批背景图片，然后将其隐藏
   2. 改进: 使用js来推迟预加载时间, 防止与页面其他内容一起加载
4. [用Ajax实现预加载](http://www.cnblogs.com/haoyijing/p/5818236.html#imgLoad4)
   1. 其实就是通过ajax请求请求图片地址. 还可以用这种方式加载css,js文件等

**35. 如果在同一个元素上绑定了两个click事件, 一个在捕获阶段执行, 一个在冒泡阶段执行. 那么当触发click条件时, 会执行几个事件? 执行顺序是什么?**

答:

 　　我在回答这个题的时候说是两个事件, 先执行捕获的后执行冒泡的. 其实是不对的.

　　绑定在目标元素上的事件是**按照绑定的顺序**执行的!!!!

　　即: **绑定在被点击元素的事件是按照代码顺序发生，其他元素通过冒泡或者捕获“感知”的事件，按照W3C的标准，先发生捕获事件，后发生冒泡事件。所有事件的顺序是：其他元素捕获阶段事件 -> 本元素代码顺序事件 -> 其他元素冒泡阶段事件 。**

　　参考: [JavaScript-父子dom同时绑定两个点击事件，一个用捕获，一个用冒泡时执行顺序](http://blog.csdn.net/qiqingjin/article/details/51387217)

**36. js中怎么实现块级作用域?**

答:

* 使用匿名函数, (立即执行函数)
  + (function(){...})()
* 或者es6
  + 块级作用域引入了两种新的声明形式,可以用它们定义一个只存在于某个语句块中的变量或常量.这两种新的声明关键字为:
    - let: 语法上非常类似于var, 但定义的变量只存在于当前的语句块中
    - const: 和let类似,但声明的是一个只读的常量

使用let代替var可以更容易的定义一个只在某个语句块中存在的局部变量,而不用担心它和函数体中其他部分的同名变量有冲突.在let语句内部用var声明的变量和在let语句外部用var声明的变量没什么差别,它们都拥有函数作用域,而不是块级作用域.

**37. 构造函数里定义function和使用prototype.func的区别？**

答：

　　　　1. 直接调用function，每一个类的实例都会拷贝这个函数，弊端就是**浪费内存**（如上）。prototype方式定义的方式，函数不会拷贝到每一个实例中，所有的实例共享prototype中的定义，节省了内存。

　　　　2. 但是如果prototype的属性是对象的话，所有实例也会共享一个对象（这里问的是函数应该不会出现这个情况），如果其中一个实例改变了对象的值，则所有实例的值都会被改变。同理的话，如果使用prototype调用的函数，一旦改变，所有实例的方法都会改变。——不可以对实例使用prototype属性，只能对类和函数用。

**38. js实现对象的深克隆**

答:

因为js中数据类型分为基本数据类型(number, string, boolean, null, undefined)和引用类型值(对象, 数组, 函数). 这两类对象在复制克隆的时候是有很大区别的. 原始类型存储的是对象的实际数据, 而对象类型存储的是对象的引用地址(对象的实际内容单独存放, 为了减少数据开销通常放在内存中). 此外, 对象的原型也是引用对象, 它把原型的属性和方法放在内存中, 通过原型链的方式来指向这个内存地址.

于是克隆也会分为两类:

1. 浅度克隆: 原始类型为值传递, 对象类型仍为引用传递
2. 深度克隆: 所有元素或属性均完全复制, 与原对象完全脱离, 也就是说所有对于新对象的修改都不会反映到原对象中

深度克隆实现:

[复制代码](javascript:void(0);)

function clone(obj){

if(typeof(obj)== 'object'){

var result = obj instanceof Array ? [] : {};

for(var i in obj){

var attr = obj[i];

result[i] = arguments.callee(attr);

}

return result;

} else {

return obj;

}

};

[复制代码](javascript:void(0);)

  参考: [JavaScript深克隆](https://www.blackglory.me/javascript-clone/)   [javascript中对象的深度克隆](http://www.cnblogs.com/rookiebob/p/3955078.html)

**六、 jQuery问题**

**七、 一些编程题**

1. var obj = {a : 1}; (function (obj) { obj = {a : 2}; })(obj); //问obj怎么变？  
答: 外部的obj不变. 因为匿名函数中obj传入参数等于是创建了一个局部变量obj, 里面的obj指向了一个新的对象 . 如果改成(function () { obj = {a : 2}; })(obj);  就会变了

2. var obj = { a:1,  func: function() { (function () { a=2; }(); }} ; obj.fun()   //a 怎么变? 匿名函数里的this 是什么?

答: obj里的a不会变. 匿名函数里的this指向全局对象window.  这等于是给window加了一个名为a的属性

　　要改变obj中a的值 , 应当:

　　(function() { this.a = 2}).call(this);

　　或者obj中定义func :    func: function() { var self = this; (function(){self.a=2;})();}

3. 要实现函数内每隔5秒调用自己这个函数，100次以后停止，怎么办

**4. 点击一个ul的五个li元素，分别弹出他们的序号，怎么做？**

想到了用闭包, 但是写错了...

一开始是这么写的

|  |  |
| --- | --- |
| 1 | //注意!!这种写法是错误的!!!　　 for(var i=0; i |

但是这样做的话, 点击所有的li都会弹出最后一个序号. 因为每个li对应的onclick事件的函数, 返回的那个函数的参数还是最后的那个i, 并没有改变. 应该是这么写

方法1 :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | for(var i=0; i<oLis.length; i++){      oLis[i].onclick = (function(j){          return function(){              alert(j);          }      })(i);  } |

这样的话, 给每个li绑定onclick事件时, 其实绑的是一个立即执行函数, 这个立即执行函数的参数是i, 因为它是立即执行的, 循环时已经把i的值赋给了li的onclick事件, 所以在外部函数里的i改变后并不会影响i的值.

另一种实现方法:(立即执行函数)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | for(var i=0; i<oLi.length; i++){      (function(j){          oLi[j].onclick = function(){              alert(j);          };      })(i);  } |

或者不用闭包

方法2:

[复制代码](javascript:void(0);)

var oLi = document.getElementsByTagName('li');

function func(obj,i){

obj.onclick = function(){

alert (i);

}

}

for(var i = 0; i<oLi.length; i++){

func(oLi[i], i);

}

[复制代码](javascript:void(0);)

方法3: 设置属性:

|  |  |
| --- | --- |
| 1  2  3  4 | var oLi = document.getElementsByTagName('li');  for(var i=0; i<oLi.length; i++){      oLi[i].setAttribute("onclick", "alert("+i+");");  } |

方法4: 设置index保存

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | for(var i=0; i<oLi.length; i++){      oLi[i].index = i;      oLi[i].onclick = function(){          alert(this.index);      }  } |

或者也可以用事件代理来做.

参考: [原生js实现点击li弹出当前的索引值(原创)](http://shiyun1013.blog.163.com/blog/static/107740362011103071022920/)

**八、js算法题**

**1. js实现数组去重怎么实现?**

**方法1**. 创建一个新的临时数组来保存数组中已有的元素

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var a = new Array(1,2,2,2,2,5,3,2,9,5,6,3);  Array.prototype.unique1 = function(){      var n = [];     //一个新的临时数组      for(var i=0; i<this.length; i++){          //如果把当前数组的第i已经保存进了临时数组, 那么跳过          if(n.indexOf(this[i]) == -1){              n.push(this[i]);          }      }      return n;  }  console.log(a.unique1()); |

**方法2.** 使用哈希表存储已有的元素

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Array.prototype.unique2 = function(){      var hash = {},          n = [];     //hash 作为哈希表, n为临时数组      for(var i=0; i<this.length; i++){          if(!hash[this[i]]){         //如果hash表中没有当前项              hash[this[i]] = true;   //存入hash表              n.push(this[i]);        //当前元素push到临时数组中          }      }      return n;  } |

**方法3**. 使用indexOf判断数组元素第一次出现的位置是否为当前位置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Array.prototype.unique3 = function(){      var n = [this[0]];      for(var i=1; i<this.length; i++)    //从第二项开始遍历      {          //如果当前数组元素在数组中出现的第一次的位置不是i          //说明是重复元素          if(this.indexOf(this[i]) == i){              n.push(this[i]);          }      }      return n;  } |

**方法4.** 先排序再去重

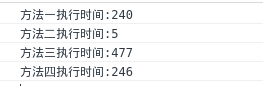
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Array.prototype.unique4 = function(){      this.sort(function(a, b){ return a - b;});      var n = [this[0]];      for(var i=1; i<this.length; i++){          if(this[i] != this[i-1]){              n.push(this[i]);          }      }      return n;  } |

第一种方法和第三种方法都使用了indexOf(), 这个函数的执行机制也会遍历数组

第二种方法使用了一个哈希表, 是最快的.

第三种方法也有一个排序的复杂度的计算.

然后做了个测试, 随机生成100万个0-1000的数组结果如下:



第三种方法总是第二种方法的将近两倍, 而第四种方法与数组的范围有关,

如果是0-100的数组



而如果是0-10000, 方法四看着就效果还不错了



而第二种方法永远是最好的, 但是是以空间换时间

全部代码如下:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | var a = [];  for(var i=0; i<1000000; i++){      a.push(Math.ceil(Math.random()\*10000));  }    Array.prototype.unique1 = function(){      var n = [];     //一个新的临时数组      for(var i=0; i<this.length; i++){          //如果把当前数组的第i已经保存进了临时数组, 那么跳过          if(n.indexOf(this[i]) == -1){              n.push(this[i]);          }      }      return n;  }    Array.prototype.unique2 = function(){      var hash = {},          n = [];     //hash 作为哈希表, n为临时数组      for(var i=0; i<this.length; i++){          if(!hash[this[i]]){         //如果hash表中没有当前项              hash[this[i]] = true;   //存入hash表              n.push(this[i]);        //当前元素push到临时数组中          }      }      return n;  }    Array.prototype.unique3 = function(){      var n = [this[0]];      for(var i=1; i<this.length; i++)    //从第二项开始遍历      {          //如果当前数组元素在数组中出现的第一次的位置不是i          //说明是重复元素          if(this.indexOf(this[i]) == i){              n.push(this[i]);          }      }      return n;  }    Array.prototype.unique4 = function(){      this.sort(function(a, b){ return a - b;});      var n = [this[0]];      for(var i=1; i<this.length; i++){          if(this[i] != this[i-1]){              n.push(this[i]);          }      }      return n;  }  var begin1 = new Date();  a.unique1();  var end1 = new Date();    var begin2 = new Date();  a.unique2();  var end2 = new Date();    var begin3 = new Date();  a.unique3();  var end3 = new Date();    var begin4 = new Date();  a.unique4();  var end4 = new Date();    console.log("方法一执行时间:" + (end1 - begin1));  console.log("方法二执行时间:" + (end2 - begin2));  console.log("方法三执行时间:" + (end3 - begin3));  console.log("方法四执行时间:" + (end4 - begin4)); |

　　参考: [js数组去重](http://www.cnblogs.com/fumj/archive/2012/09/09/2677711.html)

了解前后端分离吗（当时真不了解，但他细心地给我解释了一下，还建议我去看淘宝ued团队的文章）

用什么版本控制工具

在用mac吗

用什么来管理各种依赖

数组去重

position的几种值

css选择器优先级

伪类的用法

闭包

闭包的实现举例

浮动

清除浮动

jsonp是啥

gulp怎么用，用过啥

你觉得你还有什么闪光点我没有问到的（这个太囧了，我居然反问了面试官你觉得我刚才闪不闪？面试官呵呵呵呵）

require的读取顺序

图片轮播（虽然很多面试题都有这个，但是我自己真的没有实现过，说了一个opacity与setTimeout的组合运用，最后没有实现到面试官要求的效果，但是他说还行）

sass的伪类怎么嵌套

jsonp是啥

怎么用jsonp

发送jsonp时候的那个消息头长什么样子？（这个我直接跟他说没看过不知道）

一个文本框，需要两个控件对里面的值进行控制，一个是＋1一个是－1，要求每点击一次就有个提示框冒出来。

而且文本框是可修改的，每次修改也会冒出提示框。（这个我回答的很模糊，我说应该有个监听的事件的，但是我忘了是啥了，面试官说也行）

css定位

百度

（很惊奇的是面试我的三个面试官都是女生，简直了）

一面

css position的几种值与区别

闭包

偏向于什么语言？html、css or javascript

开发中偏于jQuery还是原生js

为什么选择原生js？

最近做过的项目

那个系统还在用吗

你是怎么学的？

看英文有障碍吗（卧槽，当时我回答了完全没有问题，不知道哪里来的自信）

二面

html5的新功能了解多少

说说离线存储

它们与cookie的区别

this的理解

怎么传入this

apply和call的区别

项目您说说

你github上面那个打字机的效果是怎么实现的（这是我fork了一位小伙伴的，但幸好我也花了一点时间弄懂了他的做法，要不然肯定囧死。插播一个很硬的广告 github.com/sidkwok/typing）

前端优化

你用过哪些优化（我说了css sprite和减少http请求，顺便引出了webpack，但没想到她也没有问）

gulp你是怎么用的

jsonp是啥

怎么跨域

为什么要跨域

怎么学习的

position几种值

具体运用是什么

display几种值

兼容性你懂多少（我就说了Ajax的还有盒子模型，但她说还有很多，但之前我说了没有系统地去学过css，都是不会的时候查文档的，估计因为这样所以她也没有追问下去）

说说你了解的框架（我说了react和vue）

react你觉得怎样（我说了一些组件化和虚拟dom树的东西）

AngularJS呢（我说没有学过，但了解过一点，我把我了解的都说了给她听）

两个的比较

为什么你会觉得AngularJS笨重？（也是自己的看法，mvc框架嘛，一整套什么都有）

jQuery还是原生js（百度的面试官都问了这个问题，我直接说对于jQuery我并不是很熟悉，因为我更喜欢研究原生js。）

为什么选择原生js（我认为要把原生吃透再去理解其他类库的话会更好一点，她说也是）

三面

（二面问面试官我表现怎样的时候她说，我可以跟你讲你已经过了，我的助理会跟你通知的。以为是hr面没想到还是技术面）

介绍自己

说说框架

比较一下框架

你打算怎么去学这些框架

听说你在浏览器兼容性这边学习的不是很好（不慌，她只是建议我这方面其实也很重要）

对自己的规划

喜欢用什么系统

来，我们聊聊人生

daocloud

一面

html语义化

h5新标签

es6

promise

promise解决了你什么问题

跨域的方法

jsonp怎么用

用过什么库

gulp的插件用过啥

webpack

为什么要打包 (我说了http请求那点事)

介绍一下react

组件化是啥

你觉的react的优点

为什么没有选择学习AngularJS

flex

响应式布局是啥

响应式布局是根据什么进行响应

css中用什么进行屏幕的判定

css中实现阴影的有什么

二面

点击a标签发生了啥

http2您说说

说说各种框架

聊聊AngularJS（说了我的理解，但不怎么了解）

那你熟悉哪个框架（正在玩React，vue也懂一点）

你给我说说你那个博客生成器（github.com/sidkwok/tech-dairy）（我知道我知道，很多广告！）

聊聊项目

聊聊模块化吧

sass你怎么用

gulp用过啥

介绍一下webpack

实现sum(2,3);sum(2,3,4);sum(2,3,4,5);(我用了比较原始的方法，if来判断)

那如果sum里面的参数不确定呢（提取arguments的长度，用for循环）

你知道伪数组吗？

ok， 再来一个数组去重吧（这太经典了吧）

* MVC理解

二面问  
项目中你解决过哪些比较困难的问题