**前言**

本文是在GitHub上看到一个大牛总结的前端常见面试题，很多问题问的都很好，很经典、很有代表性。上面没有答案，我就整理了一下，从网上找了一些相关问题的答案。里面有一部分问题的答案我也没有进行考证，不少答案都来源于网络，或许会有疏漏之处，仅供大家参考哦！(还有一部分问题答案还未整理，大家也可以自己搜索一下答案)

**1.你能描述一下渐进增强和优雅降级之间的不同吗?**

**优雅降级**：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于IE独特的盒模型布局问题，针对不同版本的IE的hack实践过优雅降级了,为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效.

**渐进增强**：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能,向页面增加无害于基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

**2.线程与进程的区别**

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。

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，从而极大地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。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

**3.说说你对语义化的理解？**

* 1：去掉或样式丢失的时候能让页面呈现清晰的结构：html本身是没有表现的，我们看到例如<h1>是粗体，字体大小2em，加粗；<strong>是加粗的，不要认为这是html的表现，这些其实html默认的css样式在起作用，所以去掉或样式丢失的时候能让页面呈现清晰的结构不是语义化的HTML结构的优点，但是浏览器都有有默认样式，默认样式的目的也是为了更好的表达html的语义，可以说浏览器的默认样式和语义化的HTML结构是不可分割的。
* 2.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。
* 3.PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对CSS的支持较弱）。
* 4.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。
* 6.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。

**4.你如何对网站的文件和资源进行优化？**

期待的解决方案包括：文件合并文件最小化/文件压缩使用CDN托管缓存的使用（多个域名来提供缓存）其他。

**5.为什么利用多个域名来提供网站资源会更有效？**

* 1.CDN缓存更方便
* 2.突破浏览器并发限制（一般每个域名建立的链接不超过6个）
* 3.Cookieless，节省带宽，尤其是上行带宽一般比下行要慢
* 4.对于UGC的内容和主站隔离，防止不必要的安全问题(上传js窃取主站cookie之类的)。正是这个原因要求用户内容的域名必须不是自己主站的子域名，而是一个完全独立的第三方域名。
* 5.数据做了划分，甚至切到了不同的物理集群，通过子域名来分流比较省事。这个可能被用的不多。

PS:关于Cookie的问题，带宽是次要的，安全隔离才是主要的。关于多域名，也不是越多越好，虽然服务器端可以做泛解释，浏览器做dns解释也是耗时间的，而且太多域名，如果要走https的话，还有要多买证书和部署的问题。

**6.请说出三种减少页面加载时间的方法。（加载时间指感知的时间或者实际加载时间）**

* 1.优化图片
* 2.图像格式的选择（GIF：提供的颜色较少，可用在一些对颜色要求不高的地方）
* 3.优化CSS（压缩合并css，如margin-top,margin-left...)
* 4.网址后加斜杠（如www.campr.com/目录，会判断这个“目录是什么文件类型，或者是目录。）
* 5.标明高度和宽度（如果浏览器没有找到这两个参数，它需要一边下载图片一边计算大小，如果图片很多，浏览器需要不断地调整页面。这不但影响速度，也影响浏览体验。当浏览器知道了高度和宽度参数后，即使图片暂时无法显示，页面上也会腾出图片的空位，然后继续加载后面的内容。从而加载时间快了，浏览体验也更好了。）
* 6.减少http请求（合并文件，合并图片）。

**7.如果你参与到一个项目中，发现他们使用Tab来缩进代码，但是你喜欢空格，你会怎么做？**

* 建议这个项目使用像EditorConfig(<http://editorconfig.org/>)之类的规范
* 为了保持一致性，接受项目原有的风格
* 直接使用VIM的retab命令

**8.请写一个简单的幻灯效果页面**

如果不使用JS来完成，可以加分。（如：纯CSS实现的幻灯片效果）

**9.你都使用哪些工具来测试代码的性能？**

Profiler,JSPerf（<http://jsperf.com/nexttick-vs-setzerotimeout-vs-settimeout>）,Dromaeo。

**10.如果今年你打算熟练掌握一项新技术，那会是什么？**

nodejs，html5，css3，less等。

**11.请谈一下你对网页标准和标准制定机构重要性的理解。**

w3c存在的意义就是让浏览器兼容性问题尽量小，首先是他们对浏览器开发者的约束，然后是对开发者的约束。

**12.什么是FOUC（无样式内容闪烁）？你如何来避免FOUC？**

FOUC(Flash Of Unstyled Content)--文档样式闪烁

<style type="text/css"media="all">@import"../fouc.css";</style>而引用CSS文件的@import就是造成这个问题的罪魁祸首。IE会先加载整个HTML文档的DOM，然后再去导入外部的CSS文件，因此，在页面DOM加载完成到CSS导入完成中间会有一段时间页面上的内容是没有样式的，这段时间的长短跟网速，电脑速度都有关系。解决方法简单的出奇，只要在<head>之间加入一个<link>或者<script>元素就可以了。

**13.doctype（文档类型）的作用是什么？你知道多少种文档类型？**

此标签可告知浏览器文档使用哪种HTML或XHTML规范。该标签可声明三种DTD类型，分别表示严格版本、过渡版本以及基于框架的HTML文档。

HTML 4.01规定了三种文档类型：Strict、Transitional以及Frameset。

XHTML 1.0规定了三种XML文档类型：Strict、Transitional以及Frameset。

Standards（标准）模式（也就是严格呈现模式）用于呈现遵循最新标准的网页，而Quirks（包容）模式（也就是松散呈现模式或者兼容模式）用于呈现为传统浏览器而设计的网页。

**14.浏览器标准模式和怪异模式之间的区别是什么？**

W3C标准推出以后，浏览器都开始采纳新标准，但存在一个问题就是如何保证旧的网页还能继续浏览，在标准出来以前，很多页面都是根据旧的渲染方法编写的，如果用的标准来渲染，将导致页面显示异常。为保持浏览器渲染的兼容性，使以前的页面能够正常浏览，浏览器都保留了旧的渲染方法（如：微软的IE）。这样浏览器渲染上就产生了Quircks mode和Standars mode，两种渲染方法共存在一个浏览器上。IE盒子模型和标准W3C盒子模型：ie的width包括：padding\border。标准的width不包括：padding\border

* 在js中如何判断当前浏览器正在以何种方式解析？

         document对象有个属性compatMode,它有两个值：BackCompat对应quirks mode，CSS1Compat对应strict mode。

**15.使用XHTML的局限有哪些？**

XHTML 与HTML的区别为：

* XHTML 元素必须被正确地嵌套。
* XHTML 元素必须被关闭。
* 标签名必须用小写字母。
* XHTML 文档必须拥有根元素。

**局限：**

所有的 XHTML 元素都必须被正确地嵌套，XHTML 必须拥有良好的结构，所有的标签必须小写，并且所有的 XHTML 元素必须被关闭。所有的 XHTML 文档必须拥有 DOCTYPE 声明，并且 html、head、title 和 body 元素必须存在。虽然代码更加的优雅，但缺少容错性，不利于快速开发。

**16.如果网页内容需要支持多语言，你会怎么做？**

下面这些问题需要考虑：

* 应用字符集的选择，选择UTF-8编码
* 语言书写习惯&导航结构
* 数据库驱动型网站

**17.data-属性的作用是什么？**

data-\* 属性用于存储页面或应用程序的私有自定义数据。data-\* 属性赋予我们在所有 HTML 元素上嵌入自定义 data 属性的能力。存储的（自定义）数据能够被页面的 JavaScript 中利用，以创建更好的用户体验（不进行 Ajax 调用或服务器端数据库查询）。

data-\* 属性包括两部分：

* 属性名不应该包含任何大写字母，并且在前缀 "data-" 之后必须有至少一个字符
* 属性值可以是任意字符串

**18.如果把HTML5看作做一个开放平台，那它的构建模块有哪些？**

<nav>,<header>,<section>,<footer>等。

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

sessionStorage和localStorage是HTML5 Web Storage API提供的，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cookies会发送到服务器端。其余两个不会。Microsoft指出InternetExplorer8增加cookie限制为每个域名50个，但IE7似乎也允许每个域名50个cookie。

* Firefox每个域名cookie限制为50个。
* Opera每个域名cookie限制为30个。
* Firefox和Safari允许cookie多达4097个字节，包括名（name）、值（value）和等号。
* Opera允许cookie多达4096个字节，包括：名（name）、值（value）和等号。
* InternetExplorer允许cookie多达4095个字节，包括：名（name）、值（value）和等号。

**20.描述下“reset”CSS文件的作用和使用它的好处。**

因为浏览器的品种很多，每个浏览器的默认样式也是不同的，所以定义一个css reset可以使各浏览器的默认样式统一。

**21.解释下浮动和它的工作原理。**

关于浮动我们需要了解，浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。要想使元素浮动，**必须为元素设置一个宽度（width）**。虽然浮动元素不是文档流之中，但是它**浮动后所处的位置依然是在浮动之前的水平方向上**。由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样，下面的元素填补原来的位置。**有些元素会在浮动元素的下方，但是这些元素的内容并不一定会被浮动的元素所遮盖，对内联元素进行定位时，这些元素会考虑浮动元素的边界，会围绕着浮动元素放置。**也可以把浮动元素想象成是被块元素忽略的元素，而内联元素会关注浮动元素的。

**22.列举不同的清除浮动的技巧，并指出它们各自适用的使用场景。**

* 1.使用空标签清除浮动。这种方法是在所有浮动标签后面添加一个空标签定义css clear:both.弊端就是增加了无意义标签。
* 2.使用overflow。给包含浮动元素的父标签添加css属性overflow:auto;zoom:1;zoom:1用于兼容IE6。
* 3.使用after伪对象清除浮动。该方法只适用于非IE浏览器。具体写法可参照以下示例。使用中需注意以下几点。一、该方法中必须为需要清除浮动元素的伪对象中设置height:0，否则该元素会比实际高出若干像素；二、content属性是必须的，但其值可以为空，content属性的值设为”.”，空亦是可以的。
* 4.浮动外部元素

此三种方法各有利弊，使用时应择优选择，比较之下第二种方法更为可取。

**23.解释下CSS sprites，以及你要如何在页面或网站中使用它。**

CSS Sprites其实就是把网页中一些背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background-repeat”，“background-position”的组合进行背景定位，background-position可以用数字能精确的定位出背景图片的位置。

**24.你最喜欢的图片替换方法是什么，你如何选择使用。**

1 <h2><span图片丢这里></span>Hello World</h2>

把span背景设成文字内容，这样又可以保证seo，也有图片的效果在上面。一般都是：alt，title，onerror。

**25.讨论CSS hacks，条件引用或者其他。**

各个浏览器都认识，这里给firefox用；

background-color:red\9;\9所有的ie浏览器可识别；

background-color:yellow\0;\0是留给ie8的+background-color:pink;+ie7定了；

\_background-color:orange;\_专门留给神奇的ie6；:root#test{background-color:purple\9;}:root是给ie9的，

@media all and(min-width:0px){#test{background-color:black\0;}}这个是老是跟ie抢着认\0的神奇的opera，必须加个\0,不然firefox，chrome，safari也都认识。

@media screen and(-webkit-min-device-pixel-ratio:0){#test{background-color:gray;}}最后这个是浏览器新贵chrome和safari的。

**26.如何为有功能限制的浏览器提供网页？你会使用哪些技术和处理方法？**

**27.如何视觉隐藏网页内容，只让它们在屏幕阅读器中可用？**

* display:none;的缺陷搜索引擎可能认为被隐藏的文字属于垃圾信息而被忽略屏幕阅读器（是为视觉上有障碍的人设计的读取屏幕内容的程序）会忽略被隐藏的文字。
* visibility:hidden;的缺陷这个大家应该比较熟悉就是隐藏的内容会占据他所应该占据物理空间3.overflow:hidden;一个比较合理的方法.texthidden{display:block;/\*统一转化为块级元素\*/overflow:hidden;width:0;height:0;}就像上面的一段CSS所展示的方法，将宽度和高度设定为0，然后超过部分隐藏，就会弥补上述一、二方法中的缺陷，也达到了隐藏内容的目的。

**28.你用过栅格系统吗？如果使用过，你最喜欢哪种？**

比如：Bootstrap，流式栅格系统，http://960.gs/，栅格系统延续美学。

**29.你用过媒体查询，或针对移动端的布局/CSS吗？**

媒体查询，就是响应式布局。通过不同的媒介类型和条件定义样式表规则。媒介查询让CSS可以更精确作用于不同的媒介类型和同一媒介的不同条件。

语法结构及用法：@media 设备名 only （选取条件） not （选取条件） and（设备选取条件），设备二{sRules}。

示例如下：

[复制代码](javascript:void(0);)

1 /\* 当浏览器的可视区域小于980px \*/

2 @media screen and （max-width： 980px） {

3 #wrap {width： 90%; margin:0 auto;}

4 #content {width： 60%;padding： 5%;}

5 #sidebar {width： 30%;}

6 #footer {padding： 8% 5%;margin-bottom： 10px;}

7 }

8 /\* 当浏览器的可视区域小于650px \*/

9 @media screen and （max-width： 650px） {

10 #header {height： auto;}

11 #searchform {position： absolute;top： 5px;right： 0;}

12 #content {width： auto; float： none; margin： 20px 0;}

13 #sidebar {width： 100%; float： none; margin： 0;}

14 }

[复制代码](javascript:void(0);)

**30.你熟悉SVG样式的书写吗？**

SVG 意为可缩放矢量图形（Scalable Vector Graphics）。

**什么是SVG？**

* SVG 指可伸缩矢量图形 (Scalable Vector Graphics)
* SVG 用来定义用于网络的基于矢量的图形
* SVG 使用 XML 格式定义图形
* SVG 图像在放大或改变尺寸的情况下其图形质量不会有所损失
* SVG 是万维网联盟的标准
* SVG 与诸如 DOM 和 XSL 之类的 W3C 标准是一个整体

书写示例如下：

1 <?xml version="1.0" standalone="no"?>

2 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

3 <svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">

4 <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red"/>

5 </svg>

**31.如何优化网页的打印样式？**

<link rel = "stylesheet" type = "text/css" media = "screen" href = "xxx.css"/>

其中media指定的属性就是设备，显示器上就是screen，打印机则是print，电视是tv，投影仪是projection。打印样式示例如下：

<link rel = "stylesheet" type = "text/css" media = "print" href = "yyy.css"/>

但打印样式表也应注意以下事项：

* 打印样式表中最好不要用背景图片，因为打印机不能打印CSS中的背景。如要显示图片，请使用html插入到页面中。
* 最好不要使用像素作为单位，因为打印样式表要打印出来的会是实物，所以建议使用pt和cm。
* 隐藏掉不必要的内容。（@print div{display:none;}）
* 打印样式表中最好少用浮动属性，因为它们会消失。如果想要知道打印样式表的效果如何，直接在浏览器上选择打印预览就可以了。

**32.在书写高效CSS时会有哪些问题需要考虑？**

* 1.样式是：从右向左的解析一个选择器；
* 2.ID最快，Universal最慢有四种类型的key selector，解析速度由快到慢依次是：ID、class、tag和universal ；
* 3.不要tag-qualify（永远不要这样做ul#main-navigation{}ID已经是唯一的，不需要Tag来标识，这样做会让选择器变慢。）；
* 4.后代选择器最糟糕（换句话说，下面这个选择器是很低效的：html body ul li a{}）；
* 5.想清楚你为什么这样写；
* 6.CSS3的效率问题（CSS3选择器（比如:nth-child）能够漂亮的定位我们想要的元素，又能保证我们的CSS整洁易读。但是这些神奇的选择器会浪费很多的浏览器资源。）；
* 7.我们知道#ID速度是最快的，那么我们都用ID，是不是很快。但是我们不应该为了效率而牺牲可读性和可维护性。

**33.使用CSS预处理器的优缺点有哪些？**

     LESS&SassLESS是受Sass启发而开发的工具，它列出了如下开发的理由：

     “为什么要开发一个Sass的替代品呢？原因很简单：首先是语法。Sass的一个关键特性是缩进式的语法，这种语法可以产生柱式外观的代码。但是你需要花费时间学习一门新的语法以及重新构建你现在的样式表。LESS给CSS带来了很多特性，使得LESS能够和CSS无缝地紧密结合在一起。因此，你可以平滑地由CSS迁移到LESS，如果你只是对使用变量或者操作感兴趣的话，你不需要学习一整门全新的语言。”

      StylusStylus相对前两者较新，可以看官方文档介绍的功能。

* 1.来自NodeJS社区，所以和NodeJS走得很近，与JavaScript联系非常紧密。还有专门JavaScript API：http://learnboost.github.io/stylus/docs/js.html；
* 2.支持Ruby之类等等框架；
* 3.更多更强大的支持和功能总结：Sass看起来在提供的特性上占有优势，但是LESS能够让开发者平滑地从现存CSS文件过渡到LESS，而不需要像Sass那样需要将CSS文件转换成Sass格式。Stylus功能上更为强壮，和js联系更加紧密。

**34.如果设计中使用了非标准的字体，你该如何去实现？**

所谓的标准字体是多数机器上都会有的，或者即使没有也可以由默认字体替代的字体。

方法：

* 用图片代替
* web fonts在线字库，如Google Webfonts，Typekit等等；<http://www.chinaz.com/free/2012/0815/269267.shtml>；
* @font-face，Webfonts(字体服务例如：Google Webfonts，Typekit等等。)

**35.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：

1 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。大体就是这样，不过浏览器还会有一些奇怪的优化。为什么从后往前匹配因为效率和文档流的解析方向。效率不必说，找元素的父亲和之前的兄弟比遍历所哟儿子快而且方便。关于文档流的解析方向，是因为现在的CSS，一个元素只要确定了这个元素在文档流之前出现过的所有元素，就能确定他的匹配情况。应用在即使html没有载入完成，浏览器也能根据已经载入的这一部分信息完全确定出现过的元素的属性。为什么是用集合主要也还是效率。基于CSS Rule数量远远小于元素数量的假设和索引的运用，遍历每一条CSS Rule通过集合筛选，比遍历每一个元素再遍历每一条Rule匹配要快得多。

**36.解释一下你对盒模型的理解，以及如何在CSS中告诉浏览器使用不同的盒模型来渲染你的布局。**

关于盒模型请看文章[CSS之布局与定位](http://www.cnblogs.com/syfwhu/p/4319236.html)。

* 请解释一下\*{box-sizing:border-box;}的作用,并且说明使用它有什么好处？

说到IE的bug，在IE6以前的版本中，IE对盒模型的解析出现一些问题，跟其它浏览器不同，将border与padding都包含在width之内。而另外一些浏览器则与它相反，是不包括border和padding的。

在我们开发的过程中会发现，有时候，如果对页面中的大区域进行设置时，将border、padding计算到width和height之内，反而更灵活。但W3C的CSS2.1规范却规定了他们并不能被包含其中。考虑到这个问题，css3中引入了一个新的属性：box-sizing，它具有“content-box”和”border-box“两个值。

1 box-sizing:content-box

当我们设置box-sizing:content-box;时，浏览器对盒模型的解释遵从我们之前认识到的W3C标准，当它定义width和height时，它的宽度不包括border和padding。

1 box-sizing:border-box

当我们设置box-sizing:border-box;时，浏览器对盒模型的解释与IE6之前的版本相同，当它定义width和height时，border和padding则是被包含在宽高之内的。内容的宽和高可以通过定义的“width”和“height”减去相应方向的“padding”和“border”的宽度得到。内容的宽和高必须保证不能为负，必要时将自动增大该元素border box的尺寸以使其内容的宽或高最小为0。

**37.请罗列出你所知道的display属性的全部值。**

display属性的值列表如下：



**38.请解释一下relative、fixed、absolute和static元素的区别？请解释一下inline和inline-block的区别？**

* 关于第一个问题请看文章[CSS之布局与定位](http://www.cnblogs.com/syfwhu/p/4319236.html) 。

**inline**:此元素会被显示为内联元素，元素前后没有换行符。

**inline-block**:行内块元素。

**39.你目前在使用哪一套CSS框架，或者在产品线上使用过哪一套？(Bootstrap,PureCSS,Foundation等等)**

* 如果有，请问是哪一套？如果可以，你如何改善CSS框架？
* 请问你有使用过CSS Flexbox或者Grid specs吗？如果有，请问在性能和效率的方面你是怎么看的？

**40.解释下事件代理。**

JavaScript事件代理则是一种简单的技巧，通过它你可以把事件处理器添加到一个父级元素上，这样就避免了把事件处理器添加到多个子级元素上。当我们需要对很多元素添加事件的时候，可以通过将事件添加到它们的父节点而将事件委托给父节点来触发处理函数。这主要得益于浏览器的事件冒泡机制。事件代理用到了两个在JavaSciprt事件中常被忽略的特性：事件冒泡以及目标元素。

1 function getEventTarget(e) {

2 e=e||window.event;

3 return e.target||e.srcElement;

4 }

**41.解释下JavaScript中this是如何工作的。**

this永远指向函数运行时所在的对象，而不是函数被创建时所在的对象。匿名函数或不处于任何对象中的函数指向window 。

1.如果是call，apply,with，指定的this是谁，就是谁。

2.普通的函数调用，函数被谁调用，this就是谁。

**42.解释下原型继承的原理。**

以下代码展示了JS引擎如何查找属性：

[复制代码](javascript:void(0);)

1 function getProperty(obj,prop) {

2 if (obj.hasOwnProperty(prop)) {

3 return obj[prop];

4 } else if (obj.\_\_proto\_\_!==null) {

5 return getProperty(obj.\_\_proto\_\_,prop);

6 } else {

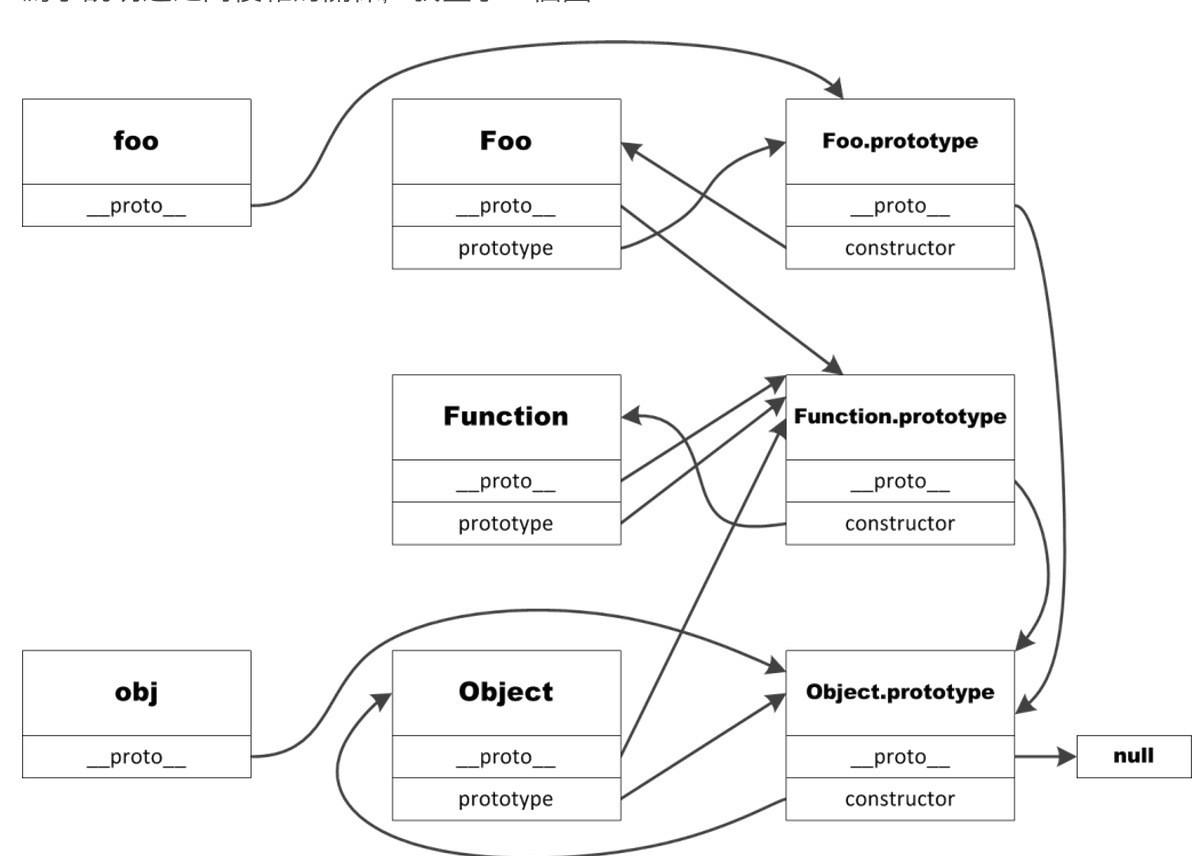
7 return undefined;

8 }

9 }

[复制代码](javascript:void(0);)

**下图展示的原（prototype）的关联：**

****

**43.你是如何测试JavaScript代码的？**

结合自己的项目经验进行讲述。（[chrome使用技巧](http://www.cnblogs.com/tonychai/p/4545812.html)）

**44.AMD vs.CommonJS？**

请看文章[JavaScript之模块化编程](http://www.cnblogs.com/syfwhu/p/4883532.html)

**45.什么是哈希表？**

散列表（也叫哈希表），是根据关键码值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。

**46.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(){}();.**

* 要做哪些改动使它变成IIFE?

因为在解析器解析全局的function或者function内部function关键字的时候，默认是认为function声明，而不是function表达式，如果你不显示告诉编译器，它默认会声明成一个缺少名字的function，并且抛出一个语法错误信息，因为function声明需要一个名字。

1 var foo = function() {

2 // doSomeThing.

3 }；

4

5 foo();

**47.描述以下变量的区别：null，undefined或undeclared？**

JavaScript的最初版本是这样区分的：null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。

但是，上面这样的区分，在实践中很快就被证明不可行。目前，null和undefined基本是同义的，只有一些细微的差别。

null表示"没有对象"，即该处不应该有值。典型用法是：

* 用来初始化一个变量，这个变量可能被赋值为一个对象。
* 用来和一个已经初始化的变量比较，这个变量可以是也可以不是一个对象。
* 当函数的参数期望是对象时，被用作参数传入。
* 当函数的返回值期望是对象时，被用作返回值传出。
* 作为对象原型链的终点。

undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。典型用法是：

* 变量被声明了，但没有赋值时，就等于undefined。
* 调用函数时，应该提供的参数没有提供，该参数等于undefined。
* 对象没有赋值的属性，该属性的值为undefined。
* 函数没有返回值时，默认返回undefined。

**该如何检测它们？**

null：表示无值；undefined：表示一个未声明的变量，或已声明但没有赋值的变量，或一个并不存在的对象属性。

==运算符将两者看作相等。如果要区分两者，要使用===或typeof运算符。

以下是不正确的用法：

1 var exp = undefined;

2

3 if (exp == undefined) {

4 alert("undefined");

5 }

exp为null时，也会得到与undefined相同的结果，虽然null和undefined不一样。注意：要同时判断undefined和null时可使用本法。

typeof返回的是字符串，有六种可能："number"、"string"、"boolean"、"object"、"function"、"undefined"。

以下是正确的用法：

1 var exp = undefined;

2

3 if(typeof(exp) == undefined) {

4 alert("undefined");

5 }

JS中如何判断null？

以下是不正确的用法：

1 var exp = null;

2

3 if(exp == null) {

4 alert("is null");

5 }

exp为undefined时，也会得到与null相同的结果，虽然null和undefined不一样。注意：要同时判断null和undefined时可使用本法。

1 var exp=null;

2

3 if(!exp) {

4 alert("is null");

5 }

如果exp为undefined或者数字零，也会得到与null相同的结果，虽然null和二者不一样。注意：要同时判断null、undefined和数字零时可使用本法。

1 var exp = null;

2

3 if(typeof(exp) == "null") {

4 alert("is null");

5 }

为了向下兼容，exp为null时，typeof总返回object。这种方式也不太好。

以下是正确的用法：

1 var exp = null;

2

3 if(!exp&&typeof(exp) != "undefined" && exp != 0) {

4 alert("is null");

5 }

**48.什么是闭包，如何使用它，为什么要使用它？**

包就是能够读取其他函数内部变量的函数。由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成“定义在一个函数内部的函数”。

所以，在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

使用闭包的注意点：

* 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。
* 闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

（关于闭包，详细了解请看[JavaScript之作用域与闭包详解](http://www.cnblogs.com/syfwhu/p/4839562.html)）

**49.请举出一个匿名函数的典型用例？**

自执行函数，用闭包模拟私有变量、特权函数等。

**50.解释“JavaScript模块模式”以及你在何时使用它。**

     如果有提到无污染的命名空间，可以考虑加分。

* 如果你的模块没有自己的命名空间会怎么样？

请看文章-[JavaScript之模块化编程](http://www.cnblogs.com/syfwhu/p/4883532.html) 和[JavaScript之命名空间模式 浅析](http://www.cnblogs.com/syfwhu/p/4885628.html)

**51.你是如何组织自己的代码？是使用模块模式，还是使用经典继承的方法？**

请看文章-[JavaScript之模块化编程](http://www.cnblogs.com/syfwhu/p/4883532.html)和[Javascript之对象的继承](http://www.cnblogs.com/syfwhu/p/4415586.html)

**52.请指出JavaScript宿主对象和原生对象的区别？**

**原生对象**

ECMA-262 把本地对象（native object）定义为“独立于宿主环境的 ECMAScript 实现提供的对象”。

“本地对象”包含哪些内容：Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError。

由此可以看出，简单来说，本地对象就是 ECMA-262 定义的类（引用类型）。

**内置对象**

ECMA-262 把内置对象（built-in object）定义为“由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现”。这意味着开发者不必明确实例化内置对象，它已被实例化了。

同样是“独立于宿主环境”。根据定义我们似乎很难分清“内置对象”与“本地对象”的区别。而ECMA-262 只定义了两个内置对象，即 Global 和 Math （它们也是本地对象，根据定义，每个内置对象都是本地对象）。如此就可以理解了。内置对象是本地对象的一种。

**宿主对象**

何为“宿主对象”？主要在这个“宿主”的概念上，ECMAScript中的“宿主”当然就是我们网页的运行环境，即“操作系统”和“浏览器”。

所有非本地对象都是宿主对象（host object），即由 ECMAScript 实现的宿主环境提供的对象。**所有的BOM和DOM都是宿主对象。**因为其对于不同的“宿主”环境所展示的内容不同。其实说白了就是，**ECMAScript官方未定义的对象都属于宿主对象，因为其未定义的对象大多数是自己通过ECMAScript程序创建的对象。**

**53.call和.apply的区别是什么？**

**call方法**:   
语法：call(thisObj，Object)  
定义：调用一个对象的一个方法，以另一个对象替换当前对象。  
说明：call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。 如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。   
**apply方法**：   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
说明：如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

对于apply和call两者在作用上是相同的，但两者在参数上有以下**区别**：  
对于第一个参数意义都一样，但对第二个参数：apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3])同时使用apply的好处是可以直接将当前函数的arguments对象作为apply的第二个参数传入。

**54.请解释Function.prototype.bind的作用？**

**55.你何时优化自己的代码？**

请看文章[JavaScript之高效编程](http://www.cnblogs.com/syfwhu/p/4438810.html) 及[JavaScript编码风格指南](http://www.cnblogs.com/syfwhu/p/4814435.html)。

**56.你能解释一下JavaScript中的继承是如何工作的吗？**

原型链等。

**57.在什么时候你会使用document.write()？**

大多数生成的广告代码依旧使用document.write()，虽然这种用法会让人很不爽。

**58.请指出浏览器特性检测，特性推断和浏览器UA字符串嗅探的区别？**

特性检测：为特定浏览器的特性进行测试，并仅当特性存在时即可应用特性。

User-Agent检测：最早的浏览器嗅探即用户代理检测，服务端（以及后来的客户端）根据UA字符串屏蔽某些特定的浏览器查看网站内容。

特性推断：尝试使用多个特性但仅验证了其中之一。根据一个特性的存在推断另一个特性是否存在。问题是，推断是假设并非事实，而且可能导致可维护性的问题。

**59.请尽可能详尽的解释AJAX的工作原理。**

请参考文章[AJAX工作原理](http://www.cnblogs.com/mingmingruyuedlut/archive/2011/10/18/2216553.html)。

**60.请解释JSONP的工作原理，以及它为什么不是真正的AJAX。**

JSONP (JSON with Padding)是一个简单高效的跨域方式，HTML中的script标签可以加载并执行其他域的javascript，于是我们可以通过script标记来动态加载其他域的资源。例如我要从域A的页面pageA加载域B的数据，那么在域B的页面pageB中我以JavaScript的形式声明pageA需要的数据，然后在 pageA中用script标签把pageB加载进来，那么pageB中的脚本就会得以执行。JSONP在此基础上加入了回调函数，pageB加载完之后会执行pageA中定义的函数，所需要的数据会以参数的形式传递给该函数。JSONP易于实现，但是也会存在一些安全隐患，如果第三方的脚本随意地执行，那么它就可以篡改页面内容，截获敏感数据。但是在受信任的双方传递数据，JSONP是非常合适的选择。

AJAX是不跨域的，而JSONP是一个是跨域的，还有就是二者接收参数形式不一样！

**61.你使用过JavaScript模板系统吗？**

如有使用过，请谈谈你都使用过哪些库，比如Mustache.js，Handlebars等等。

**62.请解释变量声明提升。**

在JS里定义的变量，存在于作用域链里，而在函数执行时会先把变量的声明进行提升，仅仅是把声明进行了提升，而其值的定义还在原来位置。示例如下：

[复制代码](javascript:void(0);)

1 var test = function() {

2 console.log(name); // 输出：undefined

3 var name = "jeri";

4 console.log(name); // 输出：jeri

5 }

6

7 test();

[复制代码](javascript:void(0);)

上述代码与下述代码等价。

[复制代码](javascript:void(0);)

1 var test = function() {

2 var name;

3 console.log(name); // 输出：undefined

4 name = "jeri";

5 console.log(name); // 输出：jeri

6 }

7

8 test();

[复制代码](javascript:void(0);)

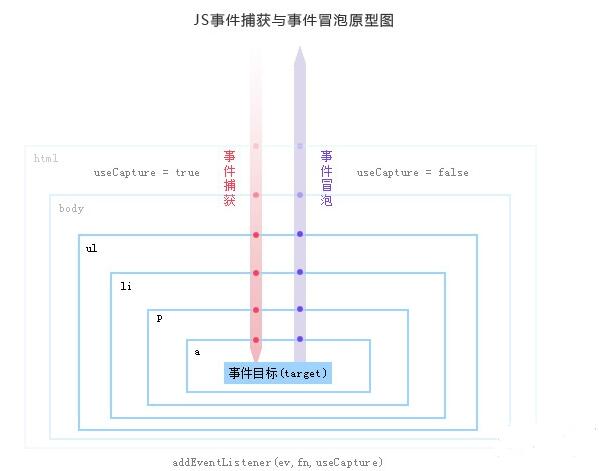
由以上代码可知，在函数执行时，把变量的声明提升到了函数顶部，而其值定义依然在原来位置。

**63.请描述下事件冒泡机制。**

冒泡型事件：事件按照从最特定的事件目标到最不特定的事件目标(document对象)的顺序触发。

捕获型事件：事件从最不精确的对象(document 对象)开始触发，然后到最精确(也可以在窗口级别捕获事件，不过必须由开发人员特别指定)。

支持W3C标准的浏览器在添加事件时用addEventListener(event,fn,useCapture)方法，基中第3个参数useCapture是一个Boolean值，用来设置事件是在事件捕获时执行，还是事件冒泡时执行。而不兼容W3C的浏览器(IE)用attachEvent()方法，此方法没有相关设置，不过IE的事件模型默认是在事件冒泡时执行的，也就是在useCapture等于false的时候执行，所以把在处理事件时把useCapture设置为false是比较安全，也实现兼容浏览器的效果。



**64."attribute"和"property"的区别是什么？**

**1. 定义**

**Property**：属性，所有的HTML元素都由HTMLElement类型表示，HTMLElement类型直接继承自Element并添加了一些属性，添加的这些属性分别对应于每个HTML元素都有下面的这5个标准特性: id,title,lang,dir,className。DOM节点是一个对象，因此，他可以和其他的JavaScript对象一样添加自定义的属性以及方法。property的值可以是任何的数据类型，对大小写敏感，自定义的property不会出现在html代码中，只存在js中。

**Attribute**：特性，区别于property，attribute只能是字符串，大小写不敏感，出现在innerHTML中，通过类数组attributes可以罗列所有的attribute。

**2. 相同之处**

标准的 DOM properties 与 attributes 是同步的。公认的（非自定义的）特性会被以属性的形式添加到DOM对象中。如，id，align，style等，这时候操作property或者使用操作特性的DOM方法如getAttribute()都可以操作属性。不过传递给getAttribute()的特性名与实际的特性名相同。因此对于class的特性值获取的时候要传入“class”。

**3. 不同之处**

1).对于有些标准的特性的操作，getAttribute与点号(.)获取的值存在差异性。如href，src，value，style，onclick等事件处理程序。  
2).href：getAttribute获取的是href的实际值，而点号获取的是完整的url，存在浏览器差异。

**65.为什么扩展JavaScript内置对象不是好的做法？**

**66.为什么扩展JavaScript内置对象是好的做法？**

**67.请指出document.onload和document.ready两个事件的区别。**

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。

**68.==和===有什么不同？**

首先，== equality 等同，=== identity 恒等。 ==， 两边值类型不同的时候，要先进行类型转换，再比较。 ===，不做类型转换，类型不同的一定不等。

先说 ===，这个比较简单。下面的规则用来判断两个值是否===相等：

* 如果类型不同，就[不相等]
* 如果两个都是数值，并且是同一个值，那么[相等]；(！例外)的是，如果其中至少一个是NaN，那么[不相等]。（判断一个值是否是NaN，只能用isNaN()来判断）
* 如果两个都是字符串，每个位置的字符都一样，那么[相等]；否则[不相等]。
* 如果两个值都是true，或者都是false，那么[相等]。
* 如果两个值都引用同一个对象或函数，那么[相等]；否则[不相等]。
* 如果两个值都是null，或者都是undefined，那么[相等]。

再说 ==，根据以下规则：

* 如果两个值类型相同，进行 === 比较。
* 如果两个值类型不同，他们可能相等。根据下面规则进行类型转换再比较：

1. 如果一个是null、一个是undefined，那么[相等]。
2. 如果一个是字符串，一个是数值，把字符串转换成数值再进行比较。
3. 如果任一值是 true，把它转换成 1 再比较；如果任一值是 false，把它转换成 0 再比较。
4. 如果一个是对象，另一个是数值或字符串，把对象转换成基础类型的值再比较。对象转换成基础类型，利用它的toString或者valueOf方法。js核心内置类，会尝试valueOf先于toString；例外的是Date，Date利用的是toString转换。非js核心的对象，令说（比较麻烦，我也不大懂）
5. 任何其他组合，都[不相等]。

**69.你如何从浏览器的URL中获取查询字符串参数。**

以下函数把获取一个key的参数。

[复制代码](javascript:void(0);)

1 function parseQueryString ( name ){

2 name = name.replace(/[\[]/,"\\\[");

3 var regexS = "[\\?&]"+name+"=([^&#]\*)";

4 var regex = new RegExp( regexS );

5 var results = regex.exec( window.location.href );

6

7 if(results == null) {

8 return "";

9 } else {

10 return results[1];

11 }

12 }

[复制代码](javascript:void(0);)

**70.请解释一下JavaScript的同源策略。**

在客户端编程语言中，如javascript和 ActionScript，同源策略是一个很重要的安全理念，它在保证数据的安全性方面有着重要的意义。同源策略规定跨域之间的脚本是隔离的，一个域的脚本不能访问和操作另外一个域的绝大部分属性和方法。那么什么叫相同域，什么叫不同的域呢？当两个域具有相同的协议, 相同的端口，相同的host，那么我们就可以认为它们是相同的域。同源策略还应该对一些特殊情况做处理，比如限制file协议下脚本的访问权限。本地的HTML文件在浏览器中是通过file协议打开的，如果脚本能通过file协议访问到硬盘上其它任意文件，就会出现安全隐患，目前IE8还有这样的隐患。

**71.请描述一下JavaScript的继承模式。**

关于继承请看文章[JavaScript之对象的继承](http://www.cnblogs.com/syfwhu/p/4415586.html)。

**72.如何实现下列代码：[1,2,3,4,5].duplicator();//[1,2,3,4,5,1,2,3,4,5]。**

**73.描述一种JavaScript中实现memoization(避免重复运算)的策略。**

**74.什么是三元表达式？“三元”表示什么意思？**

三元表达式：**? :**。**三元--三个操作对象**。

在表达式boolean-exp ? value0 : value1 中，如果“布尔表达式”的结果为true，就计算“value0”，而且这个计算结果也就是操作符最终产生的值。如果“布尔表达式”的结果为false，就计算“value1”，同样，它的结果也就成为了操作符最终产生的值。

**75.JavaScript里函数参数arguments是数组吗？**

在函数代码中，使用特殊对象 arguments，开发者无需明确指出参数名，通过使用下标就可以访问相应的参数。

arguments虽然有一些数组的性质，但其并非真正的数组，只是一个类数组对象。其并没有数组的很多方法，不能像真正的数组那样调用.jion(),.concat(),.pop()等方法。

**76.什么是"use strict";?使用它的好处和坏处分别是什么？**

在代码中出现表达式-"use strict"; 意味着代码按照**严格模式**解析，这种模式使得Javascript在更严格的条件下运行。

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。

**77.解释"chaining"。**

jQuery方法链接。直到现在，我们都是一次写一条jQuery语句（一条接着另一条）。不过，有一种名为链接（chaining）的技术，允许我们在相同的元素上运行多条jQuery命令，一条接着另一条。

提示：这样的话，浏览器就不必多次查找相同的元素。

如需链接一个动作，您只需简单地把该动作追加到之前的动作上。

**78.解释"deferreds"。**

开发网站的过程中，我们经常遇到某些耗时很长的javascript操作。其中，既有异步的操作（比如ajax读取服务器数据），也有同步的操作（比如遍历一个大型数组），它们都不是立即能得到结果的。

通常的做法是，为它们指定回调函数（callback）。即事先规定，一旦它们运行结束，应该调用哪些函数。

但是，在回调函数方面，jQuery的功能非常弱。为了改变这一点，jQuery开发团队就设计了deferred对象。

简单说，deferred对象就是jQuery的回调函数解决方案。在英语中，defer的意思是"延迟"，所以deferred对象的含义就是"延迟"到未来某个点再执行。

**79.你知道哪些针对jQuery的优化方法？**

* **1.总是从ID选择器开始继承在jQuery中最快的选择器是ID选择器，因为它直接来自于JavaScript的getElementById()方法。**

例如有一段HTML代码：

[复制代码](javascript:void(0);)

1 <div id="content">

2 <form method="post"action="#">

3 <h2>交通信号灯</h2>

4 <ul id="traffic\_light">

5 <li>

6 <input type="radio"class="on"name="light"value="red"/>红色

7 </li>

8 <li>

9 <input type="radio"class="off"name="light"value="yellow"/>黄色

10 </li>

11 <li>

12 <input type="radio"class="off"name="light"value="green"/>绿色

13 </li>

14 </ul>

15 <input class="button"id="traffic\_button"type="submit"value="Go"/>

16 </form>

17 </div>

[复制代码](javascript:void(0);)

比如需要选择红绿单选框，那么可以使用一个tag name来限制(修饰)class，如下所示：var active\_light=$(“input.on”);当然也可以结合就近的ID，如下所示：var active\_light=$(“#traffic\_light input.on”);  如果采用下面的选择器，那么效率是低效的。var traffic\_button=$(“#content.button”);因为button已经有ID了，我们可以直接使用ID选择器。如下所示：var traffic\_button=$(“#traffic\_button”);当然这只是对于单一的元素来讲。如果你需要选择多个元素，这必然会涉及到DOM遍历和循环，为了提高性能，建议从最近的ID开始继承。如下所示：var traffic\_lights=$(“#traffic\_light input”);

* **2.在class前使用tag(标签名)在jQuery中第二快的选择器是tag(标签)选择器(比如：$(“head”))。**

跟ID选择器累时，因为它来自原生的getElementsByTagName()方法。继续看刚才那段HTML代码：

在使用tag来修饰class的时候，我们需要注意以下几点：（1）不要使用tag来修饰ID，如下所示：var content=$(“div#content”);这样一来，选择器会先遍历所有的div元素，然后匹配#content。（好像jQuery从1.3.1开始改变了选择器核心后，不存在这个问题了。暂时无法考证。）（2）不要画蛇添足的使用ID来修饰ID，如下所示：var traffic\_light=$(“#content#traffic\_light”);

* **3.将jQuery对象缓存起来把jQuery对象缓存起来就是要告诉我们要养成将jQuery对象缓存进变量的习惯。**

下面是一个jQuery新手写的一段代码：

1 $("#traffic\_light input.on").bind("click",function(){});

2 $("#traffic\_light input.on").css("border","1px dashed yellow");

3 $("#traffic\_light input.on").css("background-color","orange");

4 $("#traffic\_light input.on").fadeIn("slow");

但切记不要这么做。我们应该先将对象缓存进一个变量然后再操作，如下所示：

记住，永远不要让相同的选择器在你的代码里出现多次.注：（1）为了区分普通的JavaScript对象和jQuery对象，可以在变量首字母前加上$符号。（2）上面代码可以使用jQuery的链式操作加以改善。如下所示：

[复制代码](javascript:void(0);)

1 var $active\_light = $("#traffic\_light input.on");

2

3 $active\_light.bind("click",function(){})

4 .css("border","1px dashed yellow")

5 .css("background-color","orange")

6 .fadeIn("slow");

[复制代码](javascript:void(0);)

* **4.如果你打算在其他函数中使用jQuery对象，那么你必须把它们缓存到全局环境中。**

如下代码所示：

[复制代码](javascript:void(0);)

1 // 在全局范围定义一个对象(例如:window对象)

2 window.$my = {

3 head:$("head"),

4 trafficLight:$("#trafficLight"),

5 trafficButton:$("#trafficButton")

6 };

7

8 function doSomething(){

9 // 现在你可以引用存储的结果并操作它们

10 var script=document.createElement("script");

11 $my.head.append(script);

12

13 // 当你在函数内部操作是,可以继续将查询存入全局对象中去.

14 $my.coolResults=$("#some\_ul li");

15 $my.otherResults=$("#some\_table td");

16

17 // 将全局函数作为一个普通的jquery对象去使用.

18 $my.otherResults.css("border-color","red");

19 $my.trafficLight.css("border-color","green");

20 }

21 // 你也可以在其他函数中使用它.

[复制代码](javascript:void(0);)

这里的基本思想是在内存中建立你确实想要的东西，然后更新DOM。这并不是一个jQuery最佳实践，但必须进行有效的JavaScript操作。直接的DOM操作速度很慢。例如，你想动态的创建一组列表元素，千万不要这样做,如下所示：对直接的DOM操作进行限制。

[复制代码](javascript:void(0);)

1 var top\_100\_list = [];// 假设这里是100个独一无二的字符串

2 var $mylist = $("#mylist");// jQuery选择到<ul>元素

3

4 for(var i = 0,l = top\_100\_list.length;i<l;i++) {

5 $mylist.append("<li>" + top\_100\_list[i] + "</li>");

6 }

[复制代码](javascript:void(0);)

我们应该将整套元素字符串在插入进dom中之前先全部创建好，如下所示：

1 $("#entryform input").bind("focus",function() {

2 $(this).addClass("selected");

3 }).bind("blur",function(){

4 $(this).removeClass("selected");

5 });

* **5.冒泡除非在特殊情况下,否则每一个js事件(例如:click,mouseover等.)都会冒泡到父级节点。**

当我们需要给多个元素调用同个函数时这点会很有用。代替这种效率很差的多元素事件监听的方法就是,你只需向它们的父节点绑定一次。比如,我们要为一个拥有很多输入框的表单绑定这样的行为:当输入框被选中时为它添加一个class传统的做法是，直接选中input，然后绑定focus等，如下所示：

[复制代码](javascript:void(0);)

1 $("#entryform").bind("focus",function(e) {

2 var $cell = $(e.target); // e.target捕捉到触发的目标元素

3 $cell.addClass("selected");

4 }).bind("blur",function(e) {

5 var $cell = $(e.target);

6 $cell.removeClass("selected");

7 });

[复制代码](javascript:void(0);)

当然上面代码能帮我们完成相应的任务，但如果你要寻求更高效的方法，请使用如下代码：

通过在父级监听获取焦点和失去焦点的事件，对目标元素进行操作。在上面代码中，父级元素扮演了一个调度员的角色,它可以基于目标元素绑定事件。如果你发现你给很多元素绑定了同一个事件监听,那么现在的你肯定知道哪里做错了。

* **6.推迟到$(window).load。**

jQuery对于开发者来说有一个很诱人的东西,可以把任何东西挂到$(document).ready下。尽管$(document).rady确实很有用，它可以在页面渲染时，其它元素还没下载完成就执行。如果你发现你的页面一直是载入中的状态，很有可能就是$(document).ready函数引起的。你可以通过将jQuery函数绑定到$(window).load事件的方法来减少页面载入时的cpu使用率。它会在所有的html(包括iframe)被下载完成后执行。一些特效的功能，例如拖放,视觉特效和动画,预载入隐藏图像等等，都是适合这种技术的场合。

* **7.压缩JavaScript。**

在线压缩地址:http://dean.edwards.name/packer/压缩之前，请保证你的代码的规范性，否则可能失败，导致Js错误。

* **8.尽量使用ID代替Class。**

 前面性能优化已经说过，ID选择器的速度是最快的。所以在HTML代码中，能使用ID的尽量使用ID来代替class。看下面的一个例子：

[复制代码](javascript:void(0);)

1 // 创建一个list

2 var $myList = $('#myList');

3 var myListItems = '<ul>';

4

5 for(var i = 0; i < 1000; i ++) {

6 myListItems += '<li class = "listItem'+i+'">This is a list item</li>';

7 // 这里使用的是class

8 }

9

10 myListItems += '</ul>';

11 $myList.html(myListItems);

12 // 选择每一个li

13

14 for(var i = 0; i<1000; i++) {

15 var selectedItem = $('.listItem' + i);

16 }

[复制代码](javascript:void(0);)

在上段代码中，选择每个li总共只用了61毫秒，相比class的方式，将近快了100倍。       在代码最后，选择每个li的过程中，总共用了5066毫秒，超过5秒了。接着我们做一个对比，用ID代替class：

* **9.给选择器一个上下文。**

jQuery选择器中有一个这样的选择器，它能指定上下文。jQuery(expression,context);通过它，能缩小选择器在DOM中搜索的范围，达到节省时间，提高效率。普通方式：$(‘.myDiv’)改进方式：$(‘.myDiv’,$(“#listItem”))。

* **10.慎用.live()方法（应该说尽量不要使用）。**

这是jQuery1.3.1版本之后增加的方法，这个方法的功能就是为新增的DOM元素动态绑定事件。但对于效率来说，这个方法比较占用资源。所以请尽量不要使用它。例如有这么一段代码:

[复制代码](javascript:void(0);)

1 <script type="text/javascript">

2 $(function(){

3

4 $("p").click(function(){

5 alert($(this).text());

6 });

7

8 $("button").click(function(){

9 $("<p>this is second p</p>").appendTo("body");

10 });

11 })

12 </script>

13 <body>

14 <p>this is first p</p>

15 <button>add</button>

16 </body>

[复制代码](javascript:void(0);)

运行后，你会发现新增的p元素，并没用被绑定click事件。你可以改成.live(“click”)方式解决此问题，代码如下：

[复制代码](javascript:void(0);)

1 $(function() {

2 $("p").live("click",function() {

3 // 改成live方式

4 alert($(this).text());

5 });

6

7 $("button").click(function() {

8 $("<p>this is second p</p>").appendTo("body");

9 });

10 })

[复制代码](javascript:void(0);)

但我并不建议大家这么做，我想用另一种方式去解决这个问题，代码如下：

[复制代码](javascript:void(0);)

1 $(function() {

2 $("p").click(function() {

3 alert($(this).text());

4 });

5

6 $("button").click(function() {

7 $("<p>this is second p</p>").click(function() {

8 // 为新增的元素重新绑定一次

9 alert($(this).text());

10 }).appendTo("body");

11 });

12 })

[复制代码](javascript:void(0);)

虽然我把绑定事件重新写了一次，代码多了点，但这种方式的效率明显高于live()方式，特别是在频繁的DOM操作中，这点非常明显。

**80.请解释.end()的用途。**

在官方的API上是这样描述end()方法的：“回到最近的一个"破坏性"操作之前。即，将匹配的元素列表变为前一次的状态。”；看样子好像是找到最后一次操作的元素的上一元素，在如下的例子中：html代码：

复制代码代码如下:

1 <div>测试内容1</div><div>测试内容2</div>

jQuery代码：

复制代码代码如下:

1 $('<p>新增内容</p>').appendTo('div').addClass('c1').end().addClass('c2');

复制代码代码如下:

[复制代码](javascript:void(0);)

1 <div>测试内容1

2 <p class="c1 c2">新增内容</p>

3 </div>

4 <div>测试内容2

5 <p class="c1">新增内容</p>

6 </div>

[复制代码](javascript:void(0);)

这里我就有一点不太明白了，怎么只有第一个<p>标签有两个样式，end()方法后返回的是什么,在火狐里添加了监控，得到如下结果：1.$('<p>新增内容</p>').appendTo('div')返回的是：[p,p]对象数组，即新增后的两个p标签；2.$('<p>新增内容</p>').appendTo('div').addClass('c1')返回的是：[p.c1,p.c1]对象数组，即添加了c1类样式后的p对象数组；3.$('<p>新增内容</p>').appendTo('div').addClass('c1').end()返回的是[p.c1],是第1个<div>中的<p>,在2操作中，最后“破坏”的是第2个<div>中的<p>,所以他的前一次操作的对象是第1个<div>中的<p>,返回的就是它;4.$('<p>新增内容</p>').appendTo('div').addClass('c1').end().addClass('c2')返回的仍然是第1个<div>中的<p>;现在算是有点明白了，关键是要搞清楚最后一次操作的元素的上一元素是什么。

**81.你如何给一个事件处理函数命名空间，为什么要这样做？**

任何作为type参数的字符串都是合法的；如果一个字符串不是原生的JavaScript事件名，那么这个事件处理函数会绑定到一个自定义事件上。这些自定义事件绝对不会由浏览器触发，但可以通过使用.trigger()或者.triggerHandler()在其他代码中手动触发。如果type参数的字符串中包含一个点(.)字符，那么这个事件就看做是有命名空间的了。这个点字符就用来分隔事件和他的命名空间。举例来说，如果执行.bind('click.name',handler)，那么字符串中的click是事件类型，而字符串name就是命名空间。命名空间允许我们取消绑定或者触发一些特定类型的事件，而不用触发别的事件。参考unbind()来获取更多信息。

jQuery的bind/unbind方法应该说使用很简单，而且大多数时候可能并不会用到，取而代之的是直接用click/keydown之类的事件名风格的方法来做事件绑定操作。

但假设如下情况：需要在运行时根据用户交互的结果进行不同click事件处理逻辑的绑定，因而理论上会无数次对某一个事件进行bind/unbind操作。但又希望unbind的时候只把自己绑上去的处理逻辑给释放掉而不是所有其他地方有可能的额外的同一事件绑定逻辑。这时候如果直接用.click()/.bind('click')加上.unbind('click')来进行重复绑定的话，被unbind掉的将是所有绑定在元素上的click处理逻辑，潜在会影响到该元素其他第三方的行为。

当然如果在bind的时候是显示定义了function变量的话，可以在unbind的时候提供function作为第二个参数来指定只unbind其中一个处理逻辑，但实际应用中很可能会碰到各种进行匿名函数绑定的情况。对于这种问题，jQuery的解决方案是使用事件绑定的命名空间。即在事件名称后添加.something来区分自己这部分行为逻辑范围。

比如用.bind('click.myCustomRoutine',function(){...});同样是把匿名函数绑定到click事件（你可以用自己的命名空间多次绑定不同的行为方法上去），当unbind的时候用.unbind('click.myCustomRoutine')即可释放所有绑定到.myCustomRoutine命名空间的click事件，而不会解除其他通过.bind('click')或另外的命名空间所绑定的事件行为。同时，使用命令空间还可以让你一次性unbind所有此命名空间下的自定义事件绑定，通过.unbind('.myCustomRoutine')即可。要注意的是，jQuery的命名空间并不支持多级空间。

因为在jQuery里面，如果用.unbind('click.myCustomRoutine.myCustomSubone')，解除的是命名空间分别为myCustomRoutine和myCustomSubone的两个并列命名空间下的所有click事件，而不是"myCustomRoutine下的myCustomSubone子空间"。

**82.请说出你可以传递给jQuery方法的四种不同值。**

选择器（字符串），HTML（字符串），回调函数，HTML元素，对象，数组，元素数组，jQuery对象等。

**83.什么是效果队列？**

jQuery中有个动画队列的机制。当我们对一个对象添加多次动画效果时后添加的动作就会被放入这个动画队列中，等前面的动画完成后再开始执行。可是用户的操作往往都比动画快，如果用户对一个对象频繁操作时不处理动画队列就会造成队列堆积，影响到效果。jQuery中有stop这个方法可以停止当前执行的动画，并且它有两个布尔参数，默认值都为false。第一个参数为true时会清空动画队列，第二个参数为true时会瞬间完成掉当前动画。所以，我们经常使用obj.stop(true,true)来停止动画。但是这还不够！正如jQuery文档中的说法，即使第二个参数为true，也仅仅是把当前在执行的动画跳转到完成状态。这时第一个参数如果也为true，后面的队列就会被清空。如果一个效果需要多个动画同时处理，我们仅完成其中的一个而把后面的队列丢弃了，这就会出现意料之外的结果。

**84.请指出.get()，[]，eq()的区别。**

eq:返回是一个jquery对象作用是将匹配的元素集合缩减为一个元素。这个元素在匹配元素集合中的位置变为0，而集合长度变成1。

get:是一个html对象数组作用是取得其中一个匹配的元素。num表示取得第几个匹配的元素。

**85.请指出.bind()，.live()和.delegate()的区别。**

在操纵DOM的语境中，document是根节点。现在我们可以较容易地说明.bind()、.live()和.delegate()的不同之处了。

.bind()

1 $('a').bind('click', function() {alert("That tickles!")});

这是最简单的绑定方法了。JQuery扫描文档找出所有的$(‘a’)元素，并把alert函数绑定到每个元素的click事件上。

.live()

1 $('a').live('click', function() {alert("That tickles!")});

JQuery把alert函数绑定到$(document)元素上，并使用’click’和’a’作为参数。任何时候只要有事件冒泡到document节点上，它就查看该事件是否是一个click事件，以及该事件的目标元素与’a’这一CSS选择器是否匹配，如果都是的话，则执行函数。

live方法还可以被绑定到具体的元素(或context)而不是document上，像这样：

1 $('a', $('#container')[0]).live(...);

.delegate()

1 $('#container').delegate('a', 'click', function() {alert("That tickles!")});

JQuery扫描文档查找$(‘#container’)，并使用click事件和’a’这一CSS选择器作为参数把alert函数绑定到$(‘#container’)上。任何时候只要有事件冒泡到$(‘#container’)上，它就查看该事件是否是click事件，以及该事件的目标元素是否与CCS选择器相匹配。如果两种检查的结果都为真的话，它就执行函数。

可以注意到，这一过程与.live()类似，但是其把处理程序绑定到具体的元素而非document这一根上。精明的JS’er们可能会做出这样的结论，即$('a').live()==$(document).delegate('a')，是这样吗?嗯，不，不完全是。

为什么.delegate()要比.live()好用？

基于几个原因，人们通常更愿意选用jQuery的delegate方法而不是live方法。考虑下面的例子：

1 $('a').live('click', function(){blah()}); // 或者

2 $(document).delegate('a', 'click', function(){blah()});

**速度**

后者实际上要快过前者，因为前者首先要扫描整个的文档查找所有的$(‘a’)元素，把它们存成jQuery对象。尽管live函数仅需要把’a’作为串参数传递以用做之后的判断，但是$()函数并未知道被链接的方法将会是.live()。而另一方面，delegate方法仅需要查找并存储$(document)元素。

一种寻求避开这一问题的方法是调用在$(document).ready()之外绑定的live，这样它就会立即执行。在这种方式下，其会在DOM获得填充之前运行，因此就不会查找元素或是创建jQuery对象了。

**灵活性和链能力**

live函数也挺令人费解的。想想看，它被链到$(‘a’)对象集上，但其实际上是在$(document)对象上发生作用。由于这个原因，它能够试图以一种吓死人的方式来把方法链到自身上。实际上，我想说的是，以$.live(‘a’,…)这一形式作为一种全局性的jQuery方法，live方法会更具意义一些。

**仅支持CSS选择器**

最后一点，live方法有一个非常大的缺点，那就是它仅能针对直接的CSS选择器做操作，这使得它变得非常的不灵活。

* 为什么选择.live()或.delegate()而不是.bind()？

毕竟，bind看起来似乎更加的明确和直接，难道不是吗?嗯，有两个原因让我们更愿意选择delegate或live而不是bind：

为了把处理程序附加到可能还未存在于DOM中的DOM元素之上。因为bind是直接把处理程序绑定到各个元素上，它不能把处理程序绑定到还未存在于页面中的元素之上。

如果你运行了$(‘a’).bind(…)，而后新的链接经由AJAX加入到了页面中，则你的bind处理程序对于这些新加入的链接来说是无效的。而另一方面live和delegate则是被绑定到另一个祖先节点上，因此其对于任何目前或是将来存在于该祖先元素之内的元素都是有效的。

或者为了把处理程序附加到单个元素上或是一小组元素之上，监听后代元素上的事件而不是循环遍历并把同一个函数逐个附加到DOM中的100个元素上。把处理程序附加到一个(或是一小组)祖先元素上而不是直接把处理程序附加到页面中的所有元素上，这种做法带来了性能上的好处。

**停止传播**

最后一个我想做的提醒与事件传播有关。通常情况下，我们可以通过使用这样的事件方法来终止处理函数的执行：

1 $('a').bind('click', function(e) {

2 e.preventDefault();

3 // 或者 e.stopPropagation();

4 });

不过，当我们使用live或是delegate方法的时候，处理函数实际上并没有在运行，需要等到事件冒泡到处理程序实际绑定的元素上时函数才会运行。而到此时为止，我们的其他的来自.bind()的处理函数早已运行了。

**86.请指出$和$.fn的区别，或者说出$.fn的用途。**

Jquery为开发插件提供了两个方法，分别是：

1 $.extend(obj);

2 $.fn.extend(obj);

* 1.那么这两个分别是什么意思？

$.extend(obj);是为了扩展jquery本身，为类添加新的方法。

$.fn.extend(obj);给JQUERY对象添加方法。

* 2.$.fn中的fn是什么意思，其实是prototype，即$.fn=$.prototype;

具体用法请看下面的例子：

[复制代码](javascript:void(0);)

1 $.extend({

2

3 add:function(a, b) {

4 return a+b;

5 }

6 })

7

8 $.add(5,8); // return 13

[复制代码](javascript:void(0);)

注意没有，这边的调用直接调用，前面不用任何对象。直接$.+方法名

$.fn.extend(obj)；对prototype进行扩展，为jquery类添加成员函数，jquery类的实例可以使用这个成员函数。

[复制代码](javascript:void(0);)

1 $.fn.extend({

2

3 clickwhile:function(){

4

5 $(this).click(function(){

6 alert($(this).val())

7 })

8 }

9 })

10

11 $('input').clickwhile(); // 当点击输入框会弹出该对象的Value值

[复制代码](javascript:void(0);)

注意调用时候前面是有对象的。即$('input')这么个东西。

**87.请写出一个函数实现N!的计算。N取很大的值时，该怎么办？**

使用循环、递归都能写出函数。

当N取值很大时，应该考虑把数值转化为字符串再进行运算。大数乘法再转化为大数加法运算，其具体算法应该有不少C语言实现，可以参考一下。

**88.modulo(12,5) //2 ；问题：实现满足上述结果的modulo函数。**

**89."i'm a lasagna hog".split("").reverse().join("");问题：上面的语句的返回值是什么？**

答案："goh angasal a m'i";

**90.(window.foo||(window.foo="bar"));问题：window.foo的值是什么？**

答案："bar"只有window.foo为假时的才是上面答案，否则就是它本身的值。

var foo="Hello";(function(){var bar="World";alert(foo+bar);})();alert(foo+bar);

**91.问题：上面两个alert的结果是什么？**

答案:"Hello World"和ReferenceError:bar is not defined

**92.var foo=[];foo.push(1);foo.push(2);问题：foo.length的值是什么？**

答案：2

有趣的问题：

**91.你编写过的最酷的代码是什么？其中你最自豪的是什么？**

**92.在你使用过的开发工具中，最喜欢哪个？**

**93.你有什么业余项目吗？是哪种类型的？**

**94.你最爱的IE特性是什么？**