

本文旨在加深对前端知识点的理解，资料来源于网络，由本人（博客：<http://segmentfault.com/u/trigkit4>）收集整理。

前端开发面试知识点大纲：

HTML&CSS：	对Web标准的理解、浏览器内核差异、兼容性、hack、CSS基本功：布局、盒子模型、选择器优先级及使用、HTML5、CSS3、移动端适应。
JavaScript：	数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、内存泄漏、事件机制、异步装载回调、模板引擎、Nodejs、JS
其他：	HTTP、安全、正则、优化、重构、响应式、移动端、团队协作、可维护、SEO、UED、架构、职业生涯

1.请你谈谈Cookie的弊端

cookie虽然在持久保存客户端数据提供了方便，分担了服务器存储的负担，但还是有很多局限性的。 第一：每个特定的域名下最多生成20个cookie

- 1.IE6或更低版本最多20个cookie
 - 2.IE7和之后的版本最后可以有50个cookie。
 - 3.Firefox最多50个cookie
 - 4.chrome和Safari没有做硬性限制

IE和Opera 会清理近期最少使用的cookie，Firefox会随机清理cookie。

cookie的最大大约为4096字节，为了兼容性，一般不能超过4095字节。

IE 提供了一种存储可以持久化用户数据，叫做uerData，从IE5.0就开始支持。每个数据最多128K，每个域名下最多1M。这个持久化数据放在缓存中，如果缓存没有清理，那么会一直存在。

优点：极高的扩展性和可用性

- 1.通过良好的编程，控制保存在cookie中的session对象的大小。
 - 2.通过加密和安全传输技术（SSL），减少cookie被破解的可能性。
 - 3.只在cookie中存放不敏感数据，即使被盗也不会有重大损失。
 - 4.控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。

缺点：

- 1.`Cookie` 数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉。
 - 2.安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就
 - 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。

2.浏览器本地存储

在较高版本的浏览器中，js提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。

html5中的Web Storage包括了两种存储方式：sessionStorage和localStorage。

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。

3.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

浏览器的支持除了IE 7 及以下不支持外，其他标准浏览器都完全支持(ie及FF需在web服务器里运行)，值得一提的是IE总是办好事，例如IE7、IE6中的UserData其实就是javascript本地存储的解决方案。通过简单的代码封装可以统一到所有的浏览器都支持web storage。

localStorage和sessionStorage都具有相同的操作方法，例如setItem、getItem和removeItem等

CSS 相关问题

display:none和visibility:hidden的区别？

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，就当它从来不存在。

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。

CSS中 link 和@import 的区别是？

A: (1) link属于HTML标签，而@import是CSS提供的；(2) 页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载；(3) import只在IE7以上浏览器支持，link则没有；(4) 使用@import导入的CSS文件，如果无法被加载，整个页面css将不能生效。而使用link则没有这个影响。

position的absolute与fixed共同点与不同点

A: 共同点：
1.改变行内元素的呈现方式，display被置为block；2.让元素脱离普通流，不占据空间；3.默认会覆盖到非定位元素上

B不同点：
absolute的”根元素“是可以设置的，而fixed的”根元素“固定为浏览器窗口。当你滚动网页，fixed元素与浏览器窗口之间的距离是不变的。

介绍一下CSS的盒子模型？

1) 有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 padding；

2) 盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)。

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)

* 可继承的样式: font-size font-family color, text-indent;

* 不可继承的样式: border padding margin width height ;

* 优先级就近原则，同权重情况下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,但内联比 id 要高

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。

列出**display**的值，说明他们的作用。**position**的值， **relative**和**absolute**分别是相对于谁进行定位的？

1.
block 象块类型元素一样显示。
inline 缺省值。象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.
***absolute**
生成绝对定位的元素，相对于 **static** 定位以外的第一个祖先元素进行定位。

***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，相对于其在普通流中的位置进行定位。

*** 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。

CSS3有哪些新特性？

CSS3实现圆角（border-radius），阴影（box-shadow），
对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）
transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,倾斜
增加了更多的CSS选择器 多背景 rgba
在CSS3中唯一引入的伪元素是::selection。
媒体查询，多栏布局
border-image

为什么要初始化CSS样式。

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

最简单的初始化方法就是：` {padding: 0; margin: 0;} （不建议）`

淘宝的样式初始化：

```
body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea,
body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }
h1, h2, h3, h4, h5, h6{ font-size:100%; }
address, cite, dfn, em, var { font-style:normal; }
code, kbd, pre, samp { font-family:couriernew, courier, monospace; }
```

```
small{ font-size:12px; }
ul, ol { list-style:none; }
a { text-decoration:none; }
a:hover { text-decoration:underline; }
sup { vertical-align:text-top; }
sub{ vertical-align:text-bottom; }
legend { color:#000; }
fieldset, img { border:0; }
button, input, select, textarea { font-size:100%; }
table { border-collapse:collapse; border-spacing:0; }
```

对BFC规范的理解？

BFC，块级格式化上下文，一个创建了新的BFC的盒子是独立布局的，盒子内部的子元素的样式不会影响到外面的元素。在同一个BFC中的两个毗邻的块级盒在垂直方向上依次排列。（W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行布局，以及与其他元素的关系和相互作用。）

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

html部分

说说你对语义化的理解？

- 1，去掉或者丢失样式的时候能够让页面呈现出清晰的结构
- 2，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；
- 3，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；
- 4，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。

Doctype作用？严格模式与混杂模式如何区分？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。告知浏览器以何种模式来渲染文档。
- （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。

你知道多少种Doctype文档类型？

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。

HTML与XHTML——二者有什么区别

- 区别：
- 1.所有的标记都必须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“- -”

常见兼容性问题？

* png24位的图片在ie6浏览器上出现背景，解决方案是做成PNG8.也可以引用一段脚本处理。

* 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（IE6双边距问题：在IE6下，如果对元素设置了浮动，同时又设置了margin-left或margin-right，margin值会加倍。）
#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浏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,event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.

* 解决方法:（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 {}

* 怪异模式问题:漏写DTD声明,Firefox仍然会按照标准模式来解析网页,但在IE中会触发怪异模式。为避免怪异模式给我们带来不必要的麻烦,最好养成书写DTD声明的

* 上下margin重合问题

ie和ff都存在,相邻的两个div的margin-left和margin-right不会重合,但是margin-top和margin-bottom却会发生重合。

解决方法,养成良好的代码编写习惯,同时采用margin-top或者同时采用margin-bottom。

* ie6对png图片格式支持不好(引用一段脚本处理)



解释下浮动和它的工作原理？清除浮动的技巧

浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。

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



浮动元素引起的问题和解决办法？

浮动元素引起的问题：

- (1) 父元素的高度无法被撑开，影响与父元素同级的元素
- (2) 与浮动元素同级的非浮动元素会跟随其后
- (3) 若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构

解决方法： 使用CSS中的`clear:both;`属性来清除元素的浮动可解决2、3问题，对于问题1，添加如下样式，给父元素添加`clearfix`样式：

```
.clearfix:after{content: ".";display: block;height: 0;clear: both;visibility: hidden;}
.clearfix{display: inline-block;} /* for IE/Mac */
```

清除浮动的几种方法：

- 1, 额外标签法, `<div style="clear:both;"></div>` (缺点: 不过这个办法会增加额外的标签使HTML结构看起来不够简洁。)
- 2, 使用`after`伪类

```
#parent:after{
    content:".";
    height:0;
    visibility:hidden;
    display:block;
    clear:both;
}
```

3, 浮动外部元素

4, 设置`overflow``为`hidden``或者`auto`

IE 8以下版本的浏览器中的盒模型有什么不同

IE8以下浏览器的盒模型中定义的元素的宽高不包括内边距和边框

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() //在已有的子节点前插入一个新的子节点
```

(3) 查找

```
getElementsByTagName()    //通过标签名称
```

```
getElementsByName()    //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)
```

```
getElementById()    //通过元素Id，唯一性
```

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。

* 拖拽释放(Drag and drop) API

语义化更好的内容标签 (header,nav,footer,aside,article,section)

音频、视频API(audio,video)

画布(Canvas) API

地理(Geolocation) API

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

表单控件，calendar、date、time、email、url、search

新的技术webworker，websocket，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风格；

* 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

iframe的优缺点？

1.<iframe>优点：

解决加载缓慢的第三方内容如图标和广告等的加载问题

Security sandbox

并行加载脚本

2.<iframe>的缺点：

*iframe会阻塞主页面的Onload事件；

*即时内容为空，加载也需要时间

*没有语意

如何实现浏览器内多个标签页之间的通信？

调用localStorage、cookies等本地存储方式

webSocket如何兼容低浏览器？

Adobe Flash Socket 、 ActiveX HTMLFile (IE) 、 基于 multipart 编码发送 XHR 、 基于长轮询的 XHR

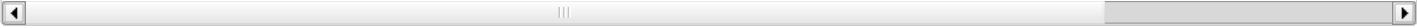
线程与进程的区别

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



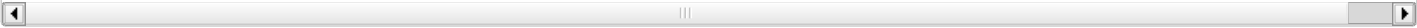
你如何对网站的文件和资源进行优化？

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

请说出三种减少页面加载时间的方法。

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）。



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

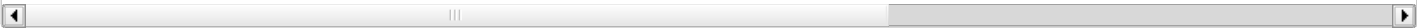
什么是 **FOUC**（无样式内容闪烁）？你如何来避免 **FOUC**？

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

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。



null和undefined的区别？

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。

当声明的变量还未被初始化时，变量的默认值为undefined。 null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

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：

- （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。

null表示"没有对象"，即该处不应该有值。典型用法是：

- （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。

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);
```

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' }

js延迟加载的方式有哪些？

`defer`和`async`、动态创建DOM方式（创建`script`，插入到DOM中，加载完毕后`callBack`）、按需异步载入js

如何解决跨域问题？

`jsonp`、`document.domain+iframe`、`window.name`、`window.postMessage`、服务器上设置代理页面

`jsonp`的原理是动态插入`script`标签

具体参见：[详解js跨域问题](#)

document.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

.call() 和 .apply() 的区别和作用？

作用：动态改变某个类的某个方法的运行环境。区别参见：[JavaScript学习总结（四）function函数部分](#)

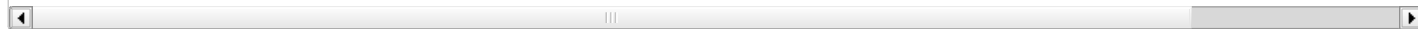
哪些操作会造成内存泄漏？

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

`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）



详见：[详解js变量、作用域及内存](#)

JavaScript中的作用域与变量声明提升？

详见：[详解JavaScript函数模式](#)

如何判断当前脚本运行在浏览器还是node环境中？

通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中

其他问题？

你遇到过比较难的技术问题是？你是如何解决的？

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？

列举IE 与其他浏览器不一样的特性？

99%的网站都需要被重构是那本书上写的？

* 网站重构：应用web标准进行设计（第2版）

什么叫优雅降级和渐进增强？

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
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

详见：[css学习归纳总结（一）](#)

WEB应用从服务器主动推送Data到客户端有那些方式？

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。

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？

你常用的开发工具是什么，为什么？

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。

你在现在的团队处于什么样的角色，起到了什么明显的作用？

你认为怎样才是全端工程师（**Full Stack developer**）？

介绍一个你最得意的作品吧？

项目中遇到什么问题？如何解决？

你的优点是什么？缺点是什么？

如何管理前端团队？

最近在学什么？能谈谈你未来**3，5**年给自己的规划吗？

你有哪些性能优化的方法？

（详情请看雅虎**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。

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** 1、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

详情：<http://segmentfault.com/blog/trigkit4/1190000000691919>

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
- （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
- （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
- （4），此时，Web服务器提供资源服务，客户端开始下载资源。

请求返回后，便进入了我们关注的前端模块

简单来说，浏览器会解析HTML生成DOM Tree，其次会根据CSS生成CSS Rule Tree，而javascript又可以根据DOM API操作DOM

详情：从输入 **URL** 到浏览器接收的过程中发生了什么事情？

平时如何管理你的项目？

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 命名以该JS功能为准的英文翻译。

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

说说最近最流行的一些东西吧？常去哪些网站？

Node.js、Mongodb、npm、MVVM、MEAN、three.js、React 。

网站：w3cfuns,sf,hacknews,CSDN,慕课，博客园，InfoQ,w3cplus等

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

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

详情：[JavaScript继承方式详解](#)

ajax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。

详情：[JavaScript学习总结（七）Ajax和Http状态字](#)

异步加载和延迟加载

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
5. 延迟加载：有些 js 代码并不是页面初始化的时候就立刻需要的，而稍后的某些情况才需要的。

前端安全问题？

(XSS, sql注入, CSRF)

CSRF：是跨站请求伪造，很明显根据刚刚的解释，他的核心也就是请求伪造，通过伪造身份提交POST和GET请求来进行跨域的攻击。

****完成CSRF需要两个步骤：****

1. 登陆受信任的网站A，在本地生成COOKIE
2. 在不登出A的情况下，或者本地COOKIE没有过期的情况下，访问危险网站B。

ie各版本和chrome可以并行下载多少个资源

IE6 两个并发，ie7升级之后的6个并发，之后版本也是6个

Firefox, chrome也是6个

javascript里面的继承怎么实现，如何避免原型链上面的对象共享

用构造函数和原型链的混合模式去实现继承，避免对象共享可以参考经典的extend()函数，很多前端框架都有封装的，就是用一个空函数当做中间变量

grunt， YUI compressor 和 google closure用来进行代码压缩的用法。

YUI Compressor 是一个用来压缩 JS 和 CSS 文件的工具，采用Java开发。

使用方法：

```
//压缩JS
java -jar yuicompressor-2.4.2.jar --type js --charset utf-8 -v src.js > packed.js
//压缩CSS
java -jar yuicompressor-2.4.2.jar --type css --charset utf-8 -v src.css > packed.css
```

详情请见：[你需要掌握的前端代码性能优化工具](#)

Flash、Ajax各自的优缺点，在使用中如何取舍？

- 1、Flash ajax对比
- Flash适合处理多媒体、矢量图形、访问机器；对CSS、处理文本上不足，不容易被搜索。
- Ajax对CSS、文本支持很好，支持搜索；多媒体、矢量图形、机器访问不足。
- 共同点：与服务器的无刷新传递消息、用户离线和在线状态、操作DOM

请解释一下 JavaScript 的同源策略。

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

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

为什么要有同源限制？

我们举例说明：比如一个黑客程序，他利用`Iframe`把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过`Javascript`读取到你的表单中中的内容，这样用户名，密码就轻松到手了。

什么是 "use strict"; ？使用它的好处和坏处分别是什么？

ECMAScript 5添加了第二种运行模式：“严格模式”（strict mode）。顾名思义，这种模式使得JavaScript在更严格的条件下运行。

设立"严格模式"的目的，主要有以下几个：

- 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。

注：经过测试IE6,7,8,9均不支持严格模式。

缺点：现在网站的JS都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被merge后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

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

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。

GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值，也就是说Get是通过地址栏来传值，而Post是通过提交表单来传值。

然而，在以下情况中，请使用 POST 请求：

无法使用缓存文件（更新服务器上的文件或数据库）

向服务器发送大量数据（POST 没有数据量限制）

发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

哪些地方会出现css阻塞，哪些地方会出现js阻塞？

js的阻塞特性：所有浏览器在下载JS的时候，会阻止一切其他活动，比如其他资源的下载，内容的呈现等等。直到JS下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。为了提高用户体验，新一代浏览器都支持并行下载JS，但是JS下载仍然会阻塞其它资源的下载（例如.图片，css文件等）。

由于浏览器为了防止出现JS修改DOM树，需要重新构建DOM树的情况，所以就会阻塞其他的下载和呈现。

嵌入JS会阻塞所有内容的呈现，而外部JS只会阻塞其后内容的显示，2种方式都会阻塞其后资源的下载。也就是说外部样式不会阻塞外部脚本的加载，但会阻塞外部脚本的执行。

CSS怎么会阻塞加载了？CSS本来是可以并行下载的，在什么情况下会出现阻塞加载了(在测试观察中，IE6下CSS都是阻塞加载)

当css后面跟着嵌入的JS的时候，该css就会出现阻塞后面资源下载的情况。而当把嵌入JS放到css前面，就不会出现阻塞的情况了。

根本原因：因为浏览器会维持html中css和js的顺序，样式表必须在嵌入的JS执行前先加载、解析完。而嵌入的JS会阻塞后面的资源加载，所以就会出现上面CSS阻塞下载的情况。

嵌入JS应该放在什么位置？

- 1、放在底部，虽然放在底部照样会阻塞所有呈现，但不会阻塞资源下载。
- 2、如果嵌入JS放在head中，请把嵌入JS放在CSS头部。
- 3、使用defer（只支持IE）
- 4、不要在嵌入的JS中调用运行时间较长的函数，如果一定要用，可以用`setTimeout`来调用

Javascript无阻塞加载具体方式

- 将脚本放在底部。<link>还是放在head中，用以保证在js加载前，能加载出正常显示的页面。<script>标签放在</body>前。
- 成组脚本：由于每个<script>标签下载时阻塞页面解析过程，所以限制页面的<script>总数也可以改善性能。适用于内联脚本和外部脚本。
- 非阻塞脚本：等页面完成加载后，再加载js代码。也就是，在window.onload事件发出后开始下载代码。（1）defer属性：支持IE4和Firefox3.5更高版本浏览器（2）动态脚本元素：文档对象模型（DOM）允许你使用js动态创建HTML的几乎全部文档内容。代码如下：

```
<script>
var script=document.createElement("script");
script.type="text/javascript";
script.src="file.js";
document.getElementsByTagName("head")[0].appendChild(script);
</script>
```

此技术的重点在于：无论在何处启动下载，文件额下载和运行都不会阻塞其他页面处理过程。即使在head里（除了用于下载文件的http链接）。

闭包相关问题？

详情请见：[详解js闭包](#)

js事件处理程序问题？

详情请见：[JavaScript学习总结（九）事件详解](#)

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次，一次解析成js语句，一次执行）。

写一个通用的事件侦听器函数？

```
// event(事件)工具集，来源: 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.removeEventListener) {
            element.removeEventListener(type, handler, false);
        } else if (element.detachEvent) {
            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;
    }
};

```

Node.js的适用场景？

高并发、聊天、实时消息推送

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。

页面重构怎么操作？

编写 CSS、让页面结构更合理化，提升用户体验，实现良好的页面效果和提升性能。

WEB应用从服务器主动推送Data到客户端有那些方式？

```
html5 websocket
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)
```

事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

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

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？

详情请见：[JavaScript学习总结（七）Ajax和Http状态字](#)

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。
2. Ajax的最大的特点是什么。

Ajax可以实现动态不刷新（局部刷新）

readyState属性 状态 有5个可取值： 0=未初始化， 1=启动 2=发送，3=接收，4=完成

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。
- 5、不容易调试。

跨域：`jsonp`、`iframe`、`window.name`、`window.postMessage`、服务器上设置代理页面

js对象的深度克隆

```
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{  
    return Obj;  
  }  
}
```

AMD和CMD 规范的区别？

详情请见：[详解JavaScript模块化开发](#)

网站重构的理解？

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

如何获取UA？

```
<script>  
  function whatBrowser() {  
    document.Browser.Name.value=navigator.appName;  
    document.Browser.Version.value=navigator.appVersion;  
    document.Browser.Code.value=navigator.appCodeName;  
    document.Browser.Agent.value=navigator.userAgent;  
  }  
</script>
```

js数组去重

以下是数组去重的三种方法：

```
Array.prototype.unique1 = 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;  
}
```

```

}

Array.prototype.unique2 = function()
{
    var n = {},r=[]; //n为hash表, r为临时数组
    for(var i = 0; i < this.length; i++) //遍历当前数组
    {
        if (!n[this[i]]) //如果hash表中没有当前项
        {
            n[this[i]] = true; //存入hash表
            r.push(this[i]); //把当前数组的当前项push到临时数组里面
        }
    }
    return r;
}

Array.prototype.unique3 = function()
{
    var n = [this[0]]; //结果数组
    for(var i = 1; i < this.length; i++) //从第二项开始遍历
    {
        //如果当前数组的第i项在当前数组中第一次出现的位置不是i,
        //那么表示第i项是重复的, 忽略掉。 否则存入结果数组
        if (this.indexOf(this[i]) == i) n.push(this[i]);
    }
    return n;
}

```

HTTP状态码

100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息
200 OK 正常返回信息
201 Created 请求成功并且服务器创建了新的资源
202 Accepted 服务器已接受请求，但尚未处理
301 Moved Permanently 请求的网页已永久移动到新位置。
302 Found 临时性重定向。
303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。
304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
401 Unauthorized 请求未授权。
403 Forbidden 禁止访问。
404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。
503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

cache-control

网页的缓存是由HTTP消息头中的“Cache-control”来控制的，常见的取值有private、no-cache、max-age、must-revalidate等，默认为private。

Expires 头部字段提供一个日期和时间，响应在该日期和时间后被认为失效。允许客户端在这个时间之前不去检查（发请求），等同max-age的效果。但是如果同时存在，则被Cache-Control的max-age覆盖。

```
Expires = "Expires" ":" HTTP-date
```

例如

```
Expires: Thu, 01 Dec 1994 16:00:00 GMT （必须是GMT格式）
```

如果把它设置为-1，则表示立即过期

Expires和max-age都可以用来指定文档的过期时间，但是二者有一些细微差别

- 1.Expires在HTTP/1.0中已经定义, Cache-Control:max-age在HTTP/1.1中才有定义, 为了向下兼容, 仅使用max-age不够;
 - 2.Expires指定一个绝对的过期时间(GMT格式),这么做会导致至少2个问题: 1)客户端和服务端时间不同步导致Expires的配置出现问题。 2)很容易在配置后忘记具体的时间;
 - 3.max-age 指定的是从文档被访问后的存活时间, 这个时间是个相对值(比如:3600s),相对的是文档第一次被请求时服务器记录的Request_time(请求时间)
 - 4.Expires指定的时间可以是相对文件的最后访问时间(Atime)或者修改时间(MTime),而max-age相对对的是文档的请求时间(Atime)
- 如果值为no-cache,那么每次都会访问服务器。如果值为max-age,则在过期之前不会重复访问服务器。

js操作获取和设置cookie

```
//创建cookie
function setCookie(name, value, expires, path, domain, secure) {
    var cookieText = encodeURIComponent(name) + '=' + encodeURIComponent(value);
    if (expires instanceof Date) {
        cookieText += '; expires=' + expires;
    }
    if (path) {
        cookieText += '; expires=' + expires;
    }
    if (domain) {
        cookieText += '; domain=' + domain;
    }
    if (secure) {
        cookieText += '; secure';
    }
    document.cookie = cookieText;
}

//获取cookie
function getCookie(name) {
    var cookieName = encodeURIComponent(name) + '=';
    var cookieStart = document.cookie.indexOf(cookieName);
    var cookieValue = null;
    if (cookieStart > -1) {
        var cookieEnd = document.cookie.indexOf(';', cookieStart);
        if (cookieEnd == -1) {
            cookieEnd = document.cookie.length;
        }
        cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));
    }
    return cookieValue;
}

//删除cookie
function unsetCookie(name) {
    document.cookie = name + "= ; expires=" + new Date(0);
}
```