

说说你对闭包的理解

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

闭包有三个特性：

1.函数嵌套函数 2.函数内部可以引用外部的参数和变量 3.参数和变量不会被垃圾回收机制回收

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 userdata，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 就可以达到目的了。

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

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。

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

cookie 和 session 的区别：

- 1、cookie 数据存放在客户的浏览器上，session 数据放在服务器上。
- 2、cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗
考虑到安全应当使用 session。
- 3、session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能
考虑到减轻服务器性能方面，应当使用 COOKIE。
- 4、单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie。
- 5、所以个人建议：
将登陆信息等重要信息存放为 SESSION
其他信息如果需要保留，可以放在 COOKIE 中

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？

- (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 IE5 以上才能识别，而 link 是 HTML 标签，无兼容问题;
- (4) link 方式的样式的权重 高于 @import 的权重.

position:absolute 和 float 属性的异同

A：共同点：

对内联元素设置`float`和`absolute`属性，可以让元素脱离文档流，并且可以设置其宽高。

B：不同点：

float 仍会占据位置，position 会覆盖文档流中的其他元素。

介绍一下 box-sizing 属性？

box-sizing 属性主要用来控制元素的盒模型的解析模式。默认值是 content-box。

content-box: 让元素维持 W3C 的标准盒模型。元素的宽度/高度由 border + padding + content 的宽度/高度决定, 设置 width/height 属性指的是 content 部分的宽/高

border-box: 让元素维持 IE 传统盒模型 (IE6 以下版本和 IE6~7 的怪异模式)。设置 width/height 属性指的是 border + padding + content

标准浏览器下, 按照 W3C 规范对盒模型解析, 一旦修改了元素的边框或内距, 就会影响元素的盒子尺寸, 就不得不重新计算元素的盒子尺寸, 从而影响整个页面的布局。

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。

position 的值, relative 和 absolute 分别是相对于谁进行定位的?

absolute

生成绝对定位的元素, 相对于最近一级的 定位不是 static 的父元素来进行定位。

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

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

XML 和 JSON 的区别？

(1). 数据体积方面。

JSON 相对于 XML 来讲, 数据的体积小, 传递的速度更快些。

(2). 数据交互方面。

JSON 与 JavaScript 的交互更加方便, 更容易解析处理, 更好的数据交互。

(3). 数据描述方面。

JSON 对数据的描述性比 XML 较差。

(4). 传输速度方面。

JSON 的速度要远远快于 XML。

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, 盒子里面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 (和布局方向有关系) 的 margin 会发生折叠。

(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。这样可以减少很多图片请求的开销, 因为请求耗时比较长; 请求虽然可以并发, 但是也有限制, 一般浏览器都是 6 个。对于未来而言, 就不需要这样做了, 因为有了 `http2`。

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.所有的标记都必须要有个相应的结束标记
- 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 “--”
- 8.图片必须有说明文字

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。

```
.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 声明的好习惯。现在可以使用 [html5](http://www.w3.org/TR/html5/single-page.html)推荐的写法: `<!doctype html>`

* 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`, 否则该元素会比实际高出若干像素;

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（内联元素）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高宽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

`getElementsByName()` //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

*没有语义

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

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。

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://jstperf.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。

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);
```

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，那么该对象的内存即可回收。

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

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？

列举 IE 与其他浏览器不一样的特性？

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。

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？

Javascript 数据推送

Commet：基于 HTTP 长连接的服务器推送技术

基于 WebSocket 的推送方案

SSE（Server-Send Event）：服务器推送数据新方式

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。

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 服务器上启动一个 DNS 查询。这能使浏览器获得请求对应的 IP 地址。

（2），浏览器与远程 Web 服务器通过 TCP 三次握手协商来建立一个 TCP/IP 连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。

（3），一旦 TCP/IP 连接建立，浏览器会通过该连接向远程服务器发送 HTTP 的 GET 请求。远程服务器找到资源并使用 HTTP 响应返回该资源，值为 200 的 HTTP 响应状态表示一个正确的响应。

（4），此时，Web 服务器提供资源服务，客户端开始下载资源。

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?

sql 注入原理

就是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串, 最终达到欺骗服务器执行恶意的 SQL 命令。

总的来说有以下几点:

- 1.永远不要信任用户的输入, 要对用户的输入进行校验, 可以通过正则表达式, 或限制长度, 对单引号和双"-进行转换等。
- 2.永远不要使用动态拼装 SQL, 可以使用参数化的 SQL 或者直接使用存储过程进行数据查询存取。
- 3.永远不要使用管理员权限的数据库连接, 为每个应用使用单独的权限有限的数据库连接。
- 4.不要把机密信息明文存放, 请加密或者 hash 掉密码和敏感的信息。

XSS 原理及防范

Xss(cross-site scripting)攻击指的是攻击者往 Web 页面里插入恶意 html 标签或者 javascript 代码。比如: 攻击者在论坛中放一个 看似安全的链接, 骗取用户点击后, 窃取 cookie 中的用户私密信息; 或者攻击者在论坛中加一个恶意表单, 当用户提交表单的时候, 却把信息传送到攻击者的服务器中, 而不是用户原本以为的信任站点。

XSS 防范方法

- 1.代码里对用户输入的地方和变量都需要仔细检查长度和对" < ", " > ", " ; ", " ' " 等字符做过滤; 其次任何内容写到页面之前都必须加以 encode, 避免不小心把 html tag 弄出来。这一个层面做好, 至少可以堵住超过一半的 XSS 攻击。
- 2.避免直接在 cookie 中泄露用户隐私, 例如 email、密码等等。
- 3.通过使 cookie 和系统 ip 绑定来降低 cookie 泄露后的危险。这样攻击者得到的 cookie 没有实际价值, 不可能拿来重放。
- 4.尽量采用 POST 而非 GET 提交表单

XSS 与 CSRF 有什么区别?

XSS 是获取信息, 不需要提前知道其他用户页面的代码和数据包。CSRF 是代替用户完成指定的动作, 需要知道其他用户页面的代码和数据包。

要完成一次 CSRF 攻击, 受害者必须依次完成两个步骤:

- 1.登录受信任网站 A, 并在本地生成 Cookie。
- 2.在不登出 A 的情况下, 访问危险网站 B。

CSRF 的防御

- 1.服务端的 CSRF 方式方法很多样, 但总的思想都是一致的, 就是在客户端页面增加伪随机数。
 - 2.使用验证码
- 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 clojure` 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 `input` 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）。
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。
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。也就是说是在不改变 UI 的情况下，对网站进行优化，在扩展的同时保持一致的 UI。

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）。

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);  
}
```

说说 TCP 传输的三次握手策略

为了准确无误地把数据送达目标处，TCP 协议采用了三次握手策略。用 TCP 协议把数据包送出去后，TCP 不会对传送后的情况置之不理，它一定会向对方确认是否成功送达。握手过程中使用了 TCP 的标志：SYN 和 ACK。发送端首先发送一个带 SYN 标志的数据包给对方。接收端收到后，回传一个带有 SYN/ACK 标志的数据包以示传达确认信息。最后，发送端再回传一个带 ACK 标志的数据包，代表“握手”结束。若在握手过程中某个阶段莫名中断，TCP 协议会再次以相同的顺序发送相同的数据包。

说说你对 Promise 的理解

依照 Promise/A+ 的定义，Promise 有四种状态：

pending: 初始状态，非 fulfilled 或 rejected.

fulfilled: 成功的操作.

rejected: 失败的操作.

settled: Promise 已被 fulfilled 或 rejected，且不是 pending

另外，fulfilled 与 rejected 一起合称 settled。

Promise 对象用来进行延迟(deferred) 和异步(asynchronous) 计算。

Promise 的构造函数

构造一个 Promise，最基本的用法如下：

```
var promise = new Promise(function(resolve, reject) {  
    if (...) { // succeed  
        resolve(result);  
    } else { // fails  
        reject(Error(errMessage));  
    }  
});
```

Promise 实例拥有 then 方法（具有 then 方法的对象，通常被称为 thenable）。它的使用方法如下：

```
promise.then(onFulfilled, onRejected)
```

接收两个函数作为参数，一个在 fulfilled 的时候被调用，一个在 rejected 的时候被调用，接收参数就是 future，onFulfilled 对应 resolve, onRejected 对应 reject。

JavaScript 垃圾回收方法

标记清除（mark and sweep）

这是 JavaScript 最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”。

垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了

引用计数(reference counting)

在低版本 IE 中经常会出现内存泄露，很多时候就是因为其采用引用计数方式进行垃圾回收。引用计数的策略是跟踪记录每个值被使用的次数，当声明了一个变量并将一个引用类型赋值给该变量的时候这个值的引用次数就加 1，如果该变量的值变成了另外一个，则这个值得引用次数减 1，当这个值的引用次数变为 0 的时候，说明没有变量在使用，这个值没法被访问了，因此可以将其占用的空间回收，这样垃圾回收器会在运行的时候清理掉引用次数为 0 的值占用的空间。

在 IE 中虽然 JavaScript 对象通过标记清除的方式进行垃圾回收，但 BOM 与 DOM 对象却是通过引用计数回收垃圾

的，也就是说只要涉及 BOM 及 DOM 就会出现循环引用问题。

谈谈性能优化问题

代码层面：避免使用 css 表达式，避免使用高级选择器，通配选择器。缓存利用：缓存 Ajax，使用 CDN，使用外部 js 和 css 文件以便缓存，添加 Expires 头，服务端配置 Etag，减少 DNS 查找等 请求数量：合并样式和脚本，使用 css 图片精灵，初始首屏之外的图片资源按需加载，静态资源延迟加载。 请求带宽：压缩文件，开启 GZIP，移动端性能优化

尽量使用 css3 动画，开启硬件加速。适当使用 touch 事件代替 click 事件。避免使用 css3 渐变阴影效果。尽可能少的使用 box-shadow 与 gradients。box-shadow 与 gradients 往往都是页面的性能杀手

什么是 Etag?

浏览器下载组件的时候，会将它们存储到浏览器缓存中。如果需要再次获取相同的组件，浏览器将检查组件的缓存时间，假如已经过期，那么浏览器将发送一个条件 GET 请求到服务器，服务器判断缓存还有效，则发送一个 304 响应，告诉浏览器可以重用缓存组件。

那么服务器是根据什么判断缓存是否还有效呢?答案有两种方式，一种是前面提到的 ETag，另一种是根据 Last-Modified

Expires 和 Cache-Control

Expires 要求客户端和服务端的时钟严格同步。HTTP1.1 引入 Cache-Control 来克服 Expires 头的限制。如果 max-age 和 Expires 同时出现，则 max-age 有更高的优先级。

Cache-Control: no-cache, private, max-age=0

ETag: abcde

Expires: Thu, 15 Apr 2014 20:00:00 GMT

Pragma: private

Last-Modified: \$now // RFC1123 format

栈和队列的区别?

栈的插入和删除操作都是在一端进行的，而队列的操作却是在两端进行的。

队列先进先出，栈先进后出。

栈只允许在表尾一端进行插入和删除，而队列只允许在表尾一端进行插入，在表头一端进行删除

栈和堆的区别?

栈区 (stack) — 由编译器自动分配释放，存放函数的参数值，局部变量的值等。

堆区 (heap) — 一般由程序员分配释放，若程序员不释放，程序结束时可能由 OS 回收。

堆 (数据结构)：堆可以被看成是一棵树，如：堆排序；

栈 (数据结构)：一种先进后出的数据结构。

关于 Http 2.0 你知道多少?

HTTP/2 引入了“服务端推 (serverpush)”的概念，它允许服务端在客户端需要数据之前就主动地将数据发送到客户端缓存中，从而提高性能。HTTP/2 提供更多的加密支持 HTTP/2 使用多路技术，允许多个消息在一个连接上同时交差。它增加了头压缩 (header compression)，因此即使非常小的请求，其请求和响应的 header 都只会占用很小比例的带宽。