keywords，是用户不查看源代码看不到的。主要作用是告诉搜索引擎本页内容是围绕哪些词展开的。因此keywords的每个词都要能在内容中找到相应匹配，才有利于排名。keywords一般不超过3个，每个关键词不宜过长，而且词语间要用英文“,”隔开。为什么用英文上文已经说过。而且，尽量将重要的关键字靠前放，因为靠后的关键字排名较差，除非你站有很高的权重。用法：<meta name=”Keywords” Content=”关键词1,关键词2,关键词3,关键词4″>

title，就是浏览器上显示的那些内容，不仅用户能看到，也能被搜索引擎检索到（搜索引擎在抓取网页时，最先读取的就是网页标题，所以title是否正确设置极其重要。）title一般不超过80个字符，而且词语间要用英文“-”隔开，因为计算机只对英语的敏感性较高，对汉语的敏感性不高。

description，和上面的keywords一样，是用户不查看源代码看不到的，而且也是对于一个网页的简要内容概况。不同的是，keywords是由几个词语的组成的，而description则是完整的一句话。description一般不超过150个字符，描述内容要和页面内容相关。

用法：<meta name=”Description” Content=”你网页的简述”>

title＞description＞keywords，

语义化：

根据内容的结构化（内容语义化），选择合适的标签（代码语义化）

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:

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。

3、写HTML代码时应注意什么？

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。

iframe的缺点：

1.会产生很多页面，不容易管理。

2.iframe框架结构有时会让人感到迷惑，如果框架个数多的话，可能会出现上下、左右滚动条，会分散访问者的注意力，用户体验度差。

3.代码复杂，无法被一些搜索引擎索引到，这一点很关键，现在的搜索引擎爬虫还不能很好的处理iframe中的内容，所以使用iframe会不利于搜索引擎优化。

4.很多的移动设备（PDA 手机）无法完全显示框架，设备兼容性差。

iframe的优点：

1.iframe能够原封不动的把嵌入的网页展现出来。

2.如果有多个网页引用iframe，那么你只需要修改iframe的内容，就可以实现调用的每一个页面内容的更改，方便快捷。

3.网页如果为了统一风格，头部和版本都是一样的，就可以写成一个页面，用iframe来嵌套，可以增加代码的可重用。

原 iframe异步加载性能优化及无阻塞加载

我们会经常使用iframes来加载第三方的内容、广告或者插件。使用iframe是因为它可以和主页面并行加载，不会阻塞主页面。

iframe会阻塞主页面的onload事件主页面和iframe共享同一个连接池

　　阻塞主页面的onload是这两个问题中最影响性能的方面。一般都是想让onload时间越早触发越好，一方面是用户体验过更重要的是google给网站的加载速度的打分：用户可以用IE和FF中Google工具栏来计时。

怎样做到iframe无阻塞加载onload？

Meebo的两个工程师(@marcuswestin and Martin Hunt)做了一个关于他们的Meebo Bar的演讲。他们使用iframe来加载一些插件，并且真正做到了无阻塞加载。对于有的开发者来说，他们的做法还比较新鲜。很赞，超级赞。

怎样做到iframe无阻塞加载onload？

<script> (function(d) { var iframe = d.body.appendChild(d.createElement('iframe')), doc = iframe.contentWindow.document; // style the iframe with some CSS iframe.style.cssText ="position:absolute;width:200px;height:100px;left:0px;"; doc.open().write('<body onload="'+ 　　'var d = document;d.getElementsByTagName(\'head\')[0].'+ 　　'appendChild(d.createElement(\'script\')).src'+'=\'\/path\/to\/file\'">'); doc.close(); //iframe onload event happens })(document);</script>

使用iframe之前需要考虑这两个缺点。如果需要使用iframe，最好是通过javascript

动态给iframe添加src属性值，

其作用是：alt属性，当图片不能正常显示时，则显示alt中的文字，表达要传递的信息，替换文字是用来替代图像而不是提供额外说明文字的除了纯装饰图片外都必须设置有意义的值，搜索引擎会重点分析。，。将title属性设置为“返回网站主页”，作用是当用户将鼠标指 向图片时，提示给用户的信息，同时也是帮助那些弱视者。

增强网站速度的方法有：

优化代码

使用CDN内容分发网络

拒绝加载额外的评论系统

减少图片大小和数量大量的视频音频

js脚本文件尽量放在一个文件中，尽量外部调用尽量减少头部代码

3PUT方法

跟POST方法很像，也是想服务器提交数据。但是，它们之间有不同。PUT指定了资源在服务器上的位置，而POST没有

4、HEAD方法

只请求页面的首部

5、DELETE方法

删除服务器上的某资源

6、OPTIONS方法

它用于获取当前URL所支持的方法。如果请求成功，会有一个Allow的头包含类似“GET,POST”这样的信息

7、TRACE方法

TRACE方法被用于激发一个远程的，应用层的请求消息回路

8、CONNECT方法

把请求连接转换到透明的TCP/IP通

<https://www.cnblogs.com/foodoir/p/5911099.html>

1.serialize()方法

　　格式：var data = $("form").serialize();

　　功能：将表单内容序列化成一个字符串。

　　这样在ajax提交表单数据时，就不用一一列举出每一个参数。只需将data参数设置为 $("form").serialize() 即可。必须有name name重复会覆盖

css sprites是什么通俗解释：CSS Sprites其实就是把网页中一些背景图片整合拼合成一张图片中，再利用DIV 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在布局盒子对象位置。

DIV CSS Sprites精灵 CSS图像拼合 CSS背景贴图定位教程案例

￼

div css sprites精灵-CSS图像拼合 CSS贴图定位网页背景素材图片拼合定位布局技术教程篇与css sprites实例篇

￼

css sprites拼合背景图片素材实现布局效果截图

一、什么是css sprites - TOP

css sprites直译过来就是CSS精灵。通常被解释为“CSS图像拼合”或“CSS贴图定位”。其实就是通过将多个图片融合到一张图里面，然后通过CSS background背景定位技术技巧布局网页背景。这样做的好处也是显而易见的，因为图片多的话，会增加http的请求，无疑促使了网站性能的减低，特别是图片特别多的网站，如果能用css sprites降低图片数量，带来的将是速度的提升。

css sprites是什么通俗解释：CSS Sprites其实就是把网页中一些背景图片整合拼合成一张图片中，再利用DIV 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在布局盒子对象位置。

二、适合与不适合DIV CSS sprites拼合布局 - TOP

1、适合：一般小图标素材

小的图标ico类素材，一般图标很小十多像素几十像素的宽度高度，这种适合拼合成一张图实现sprites background背景定位布局。多小ico太多自然加载网页时瞬间会消耗些http iis链接数，但很快加载完又会释放。

2、不适合：大图大背景

大背景一般用于网页背景，拼合时，设置为网页背景时所有背景都会显示出来。大图拼接拼合会增大图片大小，网络带宽不好的访问者访问时由于背景图大文件大会加载稍慢些，所以大图不推荐拼接拼合来使用css sprites背景定位布局。

3、sprites适合推荐小结

一般此sprites拼合布局用于局部小盒子布局不适合大背景大布局背景使用。比如小局部布局小图标背景、小导航背景等DIVCSS布局。

三、div css sprites优势与缺点劣势判断选择 - TOP

1、sprites优势：

若干小图标拼合成一张图后布局，减少http iis请求数，对于大战大流量网站来说隐形优势很显然的，从而隐形地提升了网站性能。对于大流量网站来说本来http请求数比较宝贵，使用DIV+CSS Sprites这样可以大大的提高了页面的性能，这是CSS Sprites最大的优点，也是其被广泛传播和应用的主要原因，同时也减少图片文件数目。

2、sprites缺点

在图片合并的时候，你要把多张图片有序的合理的合并成一张图片，还要留好只够的空间，防止板块内不会出现不必要的背景，如果留空间或拼合位置不合适，在布局时容易出现布局这个盒子对象时，设置背景出现拼合相邻图片，干扰图片的情况;

CSS Sprites在开发的时候比较麻烦，你要通过photoshop(PS)或其他工具测量计算每一个背景单元的精确位置，这是针线活，没什么难度，但是很繁琐;

CSS Sprites在维护的时候比较麻烦，sprites是一般双刃剑，如果页面背景有少许改动，一般就要改这张合并的图片，无需改的地方最好不要动，这样避免改动更多的css，如果在原来的地方放不下，有只能（最好）往下加图片，这样图片的字节就增加了，因为每次的图片改动都得往这个图片删除或添加内容，显得稍微繁琐，而且重新算图片的位置（尤其是这种上千px的图）也是一件颇为不爽的事情

4）、使用css sprites设置定位背景完整关键CSS代码

ul.Sprites{ margin:0 auto; border:1px solid #F00; width:300px; padding:10px;}

ul.Sprites li{ height:24px; font-size:14px;line-height:24px; text-align:left; overflow:hidden}

ul.Sprites li span{ float:left; width:17px;padding-top:5px;height:17px;

overflow:hidden;background:url(ico.png) no-repeat}

ul.Sprites li a{ padding-left:5px}

ul.Sprites li span.a1{ background-position: -62px -32px}

ul.Sprites li span.a2{ background-position: -86px -32px}

ul.Sprites li span.a3{ background-position: -110px -32px}

ul.Sprites li span.a4{ background-position: -133px -32px}

ul.Sprites li span.a5{ background-position: -158px -32px}

减少DNS查询的方法有：　缓存 DNS查找可以改善页面性能，大多数浏览器有自己的缓存系统，缓存时间又不一致，缓存时间越长，DNS保存的时间越长。

当客户端中的 DNS缓存都为空时(浏览器和操作系统都为空)， DNS查找的次数和页面中主机名的数量相同。这其中包括页面中 URL、图片、脚本文件、样式表、Flash对象等包含的主机名。减少主机名的数量可以减少DNS查找次数。

减少主机名的数量还可以减少页面中并行下载的数量。减少 DNS查找次数可以节省响应时间，但是减少并行下载却会增加响应时间。我的指导原则是 把这些页面中的内容分割成至少两部分但不超过四部分。这种结果就是在减少 DNS查找次数和保持较高程度并行下载两者之间的权衡了。

3、将css放在页面最上面，将js放在页面最下面

css放在页面最上面可以防止页面出现白屏、闪跳的现象，即减少页面的首屏出现时间。js的下载和执行会阻塞Dom树的构建（严谨地说是中断了Dom树的更新），所以script标签放在首屏范围内的HTML代码段里会截断首屏的内容。而且js中可能会对DOM节点进行操作，而这时代码是自上向下进行执行的，这样会造成js对相应的元素操作不了。所以js放在页面的最下面

4、压缩js和css

减少文件体积，去除不必要的空白符、格式符、注释（即对代码进行格式化）

5、把js和css提取出来放在外部文件中

这一条要灵活运用，把js和css提取出来放在外部文件的优点是：减少html体积，提高了js和css的复用性，提高日后的可维护性

缺点：增加了http请求，不过这一点可以通过缓存来解决

什么情况下将js和css写在页面内呢，可以分为几种情况：js和css代码比较少；这个页面不怎么会访问

6、避免重定向

重定向就是用户请求的页面被转移到了别的地方，浏览器向服务请请求一个页面，服务器告诉浏览器请求的页面已经被转移到另外一个页面，并告知另一个页面地址，浏览器就再发送请求到重定向的地址。这样会增加服务器和浏览器之间的往返次数，影响网站性能。

重定向状态码有：301永久重定向   302临时重定向。304 not modified  并不是真的重定向，它是用来告诉浏览器get请求的文件在缓存中，避免重新下载。

9、使用ajax缓存

ajax的get和post方法：

只要是浏览器的get请求，浏览器都会使用缓存，对于同一地址的请求，服务器会发送304状态码到浏览器，浏览器就会使用缓存中的数据

post的请求每次都会被执行，浏览器不会缓存

link引用的CSS会同时被加载，而@import引用的CSS 会等到页面全部被下载完再被加载。所以有时候浏览@import加载CSS的页面时开始会没有样式（就是闪烁），网速慢的还挺明显

HTML 和 XHTML 的区别简单来说，XHTML 可以认为是 XML 版本的 HTML，为符合 XML 要求，XHTML 语法上要求更严谨些。html:超文本标记语言 (Hyper Text Markup Language)

xhtml:可扩展超文本标记语言

差别1：老祖宗的差别。link属于XHTML标签，而@import完全是CSS提供的一种方式。

差别2：加载顺序的差别

差别3：兼容性的差别。由于@import是CSS2.1提出的所以老的浏览器不支持，@import只有在IE5以上的才能识别，而link标签无此问题。

差别4：使用dom控制样式时的差别。当使用javascript控制dom去改变样式的时候，只能使用link标签，因为@import不是dom可以控制的。

https://birdteam.net/3081配置etag

火狐浏览器 page seed优化图片插件

禁止图片拉伸

<img src="a.jpg" alt="Alternate Text"  style=" width:1000px; height:700px;"/>

主要分成两个部分：渲染引擎(Render Engine)和JS引擎。

渲染引擎：负责取得网页的内容(html,xml和图像等)，整理讯息(例如假如css)，以及计算网页的显示方式，然后输出到显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不同。所有网页浏览器、电子邮件客户端以及它需要编辑、显示网络内容的应用程序都需要内核。

JS引擎：解析和执行JavaScript来实现网页的动态效果。

最开始渲染引擎和JS引擎并没有区分的很明确，后来JS引擎越来越独立，内核就倾向与只指渲染引擎。

常见的浏览器内核有哪些？

Trident内核：IE，360，搜过浏览器；

Gecko内核：Netscape6及以上版本，

Presto内核：Opera

Blink内核：Opera；

Webkit内核：Safari，Chrome

WebWorker

是运行在后台的javascript，不会影响页面的性能。

在HTML页面中运行脚本时，页面状态是不可相应，直到脚本运行结束。如果使用了webworker，用户可以对页面进行点击，选取等，因为webworker运行在服务器，不会影响。

除了IE，所有浏览器都支持webworker。

WebSocket protocol 是HTML5一种新的协议。它实现了浏览器与服务器全双工通信(full-duple)。一开始的握手需要借助HTTP请求完成。

现很多网站为了实现即时通讯，所用的技术都是轮询(polling)。轮询是在特定的的时间间隔（如每1秒），由浏览器对服务器发出HTTP request，然后由服务器返回最新的数据给客户端的浏览器。这种传统的HTTP request 的模式带来很明显的缺点 – 浏览器需要不断的向服务器发出请求，然而HTTP request 的header是非常长的，里面包含的有用数据可能只是一个很小的值，这样会占用很多的带宽。

在 WebSocket API，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。在此WebSocket 协议中，为我们实现即时服务带来了两大好处：

1. Header

互相沟通的Header是很小的-大概只有 2 Bytes

2. Server Push

服务器的推送，服务器不再被动的接收到浏览器的request之后才返回数据，而是在有新数据时就主动推送给浏览器。

在实现websocket连线过程中，需要通过浏览器发出websocket连线请求，然后服务器发出回应，这个过程通常称为“握手” (handshaking)。

WebWorker

是运行在后台的javascript，不会影响页面的性能。

在HTML页面中运行脚本时，页面状态是不可相应，直到脚本运行结束。如果使用了webworker，用户可以对页面进行点击，选取等，因为webworker运行在服务器，不会影响。

除了IE，所有浏览器都支持webworker。

WebStorage

可以在客户端存储数据，类似于HTML4的cookie，但比cookie更强大。

Web Storage又分为两种：

 sessionStorage localStorage

从字面意思就可以很清楚的看出来，sessionStorage将数据保存在session中，浏览器关闭也就没了；而localStorage则一直将数据保存在客户端本地，除非主动删除数据，否则数据是永远不会过期的:localStorage和sessionStorage一样都是用来存储客户端临时信息的对象。

局限

1、浏览器的大小不统一，并且在IE8以上的IE版本才支持localStorage这个属性

2、目前所有的浏览器中都会把localStorage的值类型限定为string类型，这个在对我们日常比较常见的JSON对象类型需要一些转换

3、localStorage在浏览器的隐私模式下面是不可读取的

4、localStorage本质上是对字符串的读取，如果存储内容多的话会消耗内存空间，会导致页面变卡

5、localStorage不能被爬虫抓取到

6localStorage的使用也是遵循同源策略的，所以不同的网站直接是不能共用相同的localStorage

setItem

存储数据【增】

getItem 读取数据【查单个】

removeItem

删除某个数据【删单个】

clear

删除全部数据【删全部】

length localStorage存储变量的个数【计算数据总数】

key 读取第i个数据的名字或称为键值(从0开始计数)

valueOf 获取所有存储的数据【查全部】

他们均只能存储字符串类型的对象（虽然规范中可以存储其他原生类型的对象，但是目前为止没有浏览器对其进行实现）。

1.如何为网页添加离线存储功能

答：创建一个和html同名的manifest文件，比如页面为index.html，那么可以建一个index.manifest的文件，注意，这里的扩展名是任意的。然后给index.html的html标签添加属性即可

1.如何为网页添加离线存储功能

<https://blog.csdn.net/dj0379/article/details/53350309>

答：创建一个和html同名的manifest文件，比如页面为index.html，那么可以建一个index.manifest的文件，注意，这里的扩展名是任意的。然后给index.html的html标签添加属性即可

Cookie是由服务器端生成，发送给User-Agent（一般是浏览器），浏览器会将Cookie的key/value保存到某个目录下的文本文件内，下次请求同一网站时就发送该Cookie给服务器（前提是浏览器设置为启用cookie）。

主动删除数据localStorage

function del\_click(){

 localStorage.clear();

 $("pstauts").style.display = "block";

 $("pstauts").innerHTML = "全部数据已经

）作为语义化标签，section 应用的典型场景有文章的章节、标签对话框中的标签页、或者论文中有编号的部分。一般来说，当元素内容明确地出现在文档大纲中时，section 就是适用的。

3）对于article标签来说，无论从结构上还是内容上来说，article 本身就是独立的、完整的。有个最简单的判断方法是看这段内容脱离了所在的语境，是否还是完整的、独立的，如果是，则应该用article标签。

4）div section article ，语义是从无到有，逐渐增强的。div 无任何语义，仅仅用作样式化或者脚本化，对于一段主题性的内容，则就适用 section，而假如这段内容可以脱离上下文，作为完整的独立存在的一段内容，则就适用 article。原则上来说，能使用 article 的时候，也是可以使用 section 的，但是实际上，假如使用 article 更合适，那么就不要使用 section

html5表单控件

2）.required（HTML5）:规定控件内容是必填的。

（3）.form（HTML5）:规定输入域所属的一个或多个表单。

（4）.autofocus（HTML5）：规定在页面加载时，域自动地获得焦点。

geoglocation位置定位

web标准简单来说可以分为结构、表现和行为。其中结构主要是有HTML标签组成。或许通俗点说，在页面body里面我们写入的标签都是为了页面的结构。表现即指css样式表，通过css可以是页面的结构标签更具美感。行为是指页面和用户具有一定的交互，同时页面结构或者表现发生变化，主要是有js组成。

web标准一般是将该三部分独立分开，使其更具有模块化。但一般产生行为时，就会有结构或者表现的变化，也使这三者的界限并不那么清晰。

W3C对web标准提出了规范化的要求，也就是在实际编程中的一些代码规范：包含如下几点

1.对于结构要求：（标签规范可以提高搜索引擎对页面的抓取效率，对SEO很有帮助）

1）标签字母要小写

2）标签要闭合

3）标签不允许随意嵌套

2.对于css和js来说

1）尽量使用外链css样式表和js脚本。是结构、表现和行为分为三块，符合规范。同时提高页面渲染速度，提高用户的体验。

2）样式尽量少用行间样式表，使结构与表现分离，标签的id和class等属性命名要做到见文知义，标签越少，加载越快，用户体验提高，代码维护简单，便于改版

3）不需要变动页面内容，便可提供打印版本而不需要复制内容，提高网站易用性。

不同浏览器对HTML标记所具有的内外边距属性具有不同的定义。

因此如果想消除这种差距，应该在相应的CSS部分加入以下CSS代码：

\*{margin:0px;padding:0px;}

Margin不一致的问题

DIV居中问题https://zhidao.baidu.com/question/1495498823243622619.html?entrytime=1541553534249

如果你的页面添加了<!DOCTYPE html>那么，那么就等同于开启了标准模式浏览器就得老老实实的按照W3C的标准解析渲染页面，它的目的是要告诉标准通用标记语言解析器，它应该使用什么样的文档类型定义（DTD）来解析文档。

严格模式：又称标准模式，是指浏览器按照 W3C 标准解析代码。

混杂模式：又称怪异模式或兼容模式，是指浏览器用自己的方式解析代码。

HTML5 没有严格和混杂之分

什么是FOUC？如何避免FOUC？

如果使用import方法对CSS进行导入,会导致某些页面在Windows 下的Internet Explorer出现一些奇怪的现象:以无样式显示页面内容的瞬间闪烁,这种现象称之为文档样式短暂失效(Flash of Unstyled Content),简称为FOUC.原因大致为： 1，使用import方法导入样式表。 2，将样式表放在页面底部 3，有几个样式表，放在html结构的不同位置。其实原理很清楚：当样式表晚于 结构性html 加载，当加载到此样式表时，页面将停止之前的渲染。此样式表被下载和解析后，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标签将样式表放在文档HEAD中更多

①如果display设置为none，那position和float设置成花都没用，页面上是渲染不出此元素的，这种情况下display比较厉害；

②如果display不为none，假设为inline，然后设置float不为none，如 float: left;

   此时经浏览器计算后是属性是float: left; display: block; 这种情况下float会强制更改元素display属性；

③如果display不为none，假设为inline，然后设置position: absolute/fixed; float: left;

   此时经浏览器计算后是属性是 position: absolute; float: none; display: block;

   这种情况下position会比float和display厉害，然后强制改变float为none、display为block

<https://www.cnblogs.com/dolphinX/p/3508869.htmlBFC>

css初始化样式 html元素的默认边距，统一字体样式 清除table默认边框样式 列表元素默认样式设置

超链接下划线

border-box 告诉浏览器去理解你设置的边框和内边距的值是包含在width内的。也就是说，如果你将一个元素的width设为100px,那么这100px会包含其它的border和padding，内容区的实际宽度会是width减去border + padding的计算值。大多数情况下这使得我们更容易的去设定一个元素的宽高。

content-box  是默认值。如果你设置一个元素的宽为100px，那么这个元素的内容区会有100px宽，并且任何边框和内边距的宽度都会被增加到最后绘制出来的元素宽度中。

css3新增属性选择器

<https://www.cnblogs.com/dreamingbaobei/p/5062998.html>

<https://blog.csdn.net/kf_qyl/article/details/79005082>

但是IE8以及更低版本的IE浏览器不支持css3伪类，所以在使用时要是涉及到布局等意象全局的样式，应该多考虑一下。

注释：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。https://blog.csdn.net/hello\_world\_csdn/article/details/54710951

如此inline block边距方法

去掉HTML中的空格

使用margin负值

.space a { display: inline-block; margin-right: -3px; }

由于外部环境的不确定性，以及最后一个元素多出的父margin值等问题，这个方法不适合大规模使用。

在HTML5中，我们直接：

<div class="space"> < a href="##">惆怅 < a href="##">淡定 < a href="##">热血 </div>

基本上可以解决大部分浏览器下inline-block元素之间的间距(IE7等浏览器有时候会有1像素的间距)。不过有个浏览器，就是Chrome, 其默认有最小字体大小限制，因为，考虑到兼容性，我们还需要添加：

.space {

font-size: 0;

-webkit-text-size-adjust:none;

}

letter-spacing 属性增加或减少字符间的空白（字符间距）。

.space { letter-spacing: -3px; } .space a { letter-spacing: 0; }

Opera浏览器下有蛋疼的问题：最小间距1像素，然后，letter-spacing再小就还原了。

letter-spacing添加字母之间的空白，而word-spacing添加每个单词之间的空白。

CSS的浮动，更准确的说是水平浮动

元素的水平浮动，意味着元素只能左右浮动，而不能上下浮动

一个浮动的元素会尽量向左或向右移动，直到它的外边框遇到了包含框或另一个浮动的边框为止

浮动元素之后的元素将围绕它（此处指的是图像布局）

浮动元素之前的元素不受影响

浮动特性：

1、块级元素浮动将并排显示，不再独占一行

2、内联样式浮动将支持设置宽高，不仅支持margin-left(right)，而且开始支持margin-top(bottom)

3、浮动元素默认内容撑开宽度

4、浮动元素脱离正常的文档流

5、浮动元素提升半层级

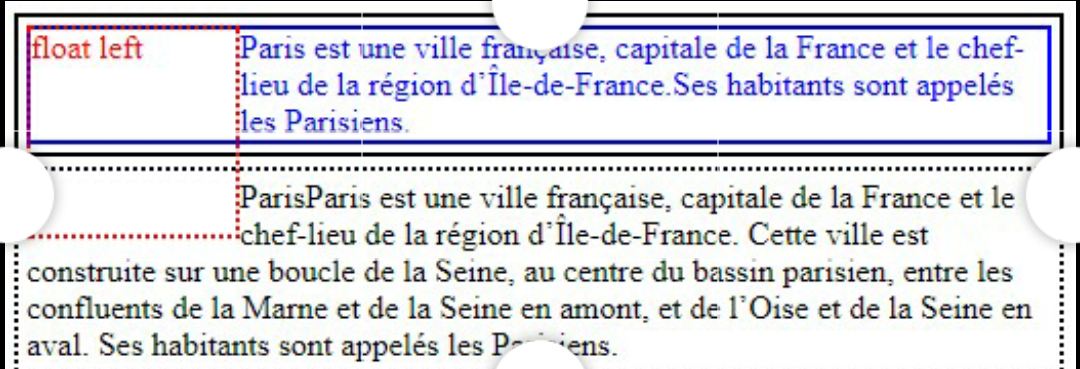
float在绝对定位和display为none时不会被应用。

float就是隐性的把内联元素转化为块元素，这是对内部的特性就是有物理特性，但是他不占据一行。对外是内联元素的属性。他有个坏处就是会影响兄弟元素。相当于：display:inline-block;

对于块元素float之后的改变（其实和内联元素一样，相当于变成了：inline-block元素，最明显的是宽度自适应发生首先div的默认宽度是width:auto;意思是自动调整宽度.

在不带float的情况下,div的宽度会自动调整至最大化,而在带float的情况下则正好相反,它会自动调整至最小化.

第三div带float时,只要宽度不超过网页宽度,则其它元素包括div都要环绕它.或者也可以认为此时的带float的div是一个特殊的 padding



BFC会阻止垂直外边距（margin-top、margin-bottom）折叠

按照BFC的定义，只有同属于一个BFC时，两个元素才有可能发生垂直Margin的重叠，这个包括相邻元素，嵌套元素，只要他们之间没有阻挡(例如边框，非空内容，padding等)就会发生margin重叠。

因此要解决margin重叠问题，只要让它们不在同一个BFC就行了，但是对于两个相邻元素来说，意义不大，没有必要给它们加个外壳，但是对于嵌套元素来说就很有必要了，只要把父元素设为BFC就可以了。这样子元素的margin就不会和父元素的margin发生重叠了。

BFC不会重叠浮动元素

BFC可以包含浮动

我们可以利用BFC的第三条特性来“清浮动”，这里其实说清浮动已经不再合适，应该说包含浮动。也就是说只要父容器形成BFC就可

IE6和IE7不支持，inherit

闭包会造成内存泄漏，就是内存被占用，无法腾出空间做其他的操作。这就牵扯到了浏览器的垃圾回收机制，

浏览器的垃圾回收机制分为两种；

其一：标记清除。具体就是当浏览器运行js时，会给所有的变量，或者占用空间内存的其它东西全部加上标记（就是一个记号，不需要过多研究），执行中当变量被引用到，该变量的标记会被去掉，到最后还存有标记的变量说明已经没有用了，浏览器会把它清除掉，并且收回占用的空间，浏览器会不断地重复执行，所以当变量不用时最终还是会被清除掉，，。而闭包里面的变量受到保护，处于一直被占用状态，也就标记不上了，在内存中也就一直占用其二：引用计数。引用计数的含义是跟踪记录每个值被引用的次数，当声明一个变量并将一个引用类型的值赋给该变量时，这个时候的引用类型的值就会是引用次数+1了。如果同一个值又被赋给另外一个变量，则该值的引用次数又+1。

相反如果包含这个值的引用的变量又取得另外一个值，即被重新赋了值，那么这个值的引用就减一。当这个值的引用次数编程0时，表示没有用到这个值，这个值也无法访问，因此环境就会收回这个值所占用的内存空间回收。这样，当垃圾收集器下次再运行时，它就会释放引用次数为0的值所占用的内存。

JavaScript没有私有成员的概念，所有对象属性都是公共的。不过JavaScript有私有变量的概念。任何在函数中定义的变量，都可以认为是私有变量，因为不能在函数外面访问这些变量。

这种有权访问私有变量和私有函数的公有方法被称为特权方法(privileged method)。

https://blog.csdn.net/iteye\_6312/article/details/82451066私有变量和静态私有变量区别

IE中的attachEvent中的this总是指向全局对象Window

new操作符调用对象时会经历四个步骤

创建一个新对象

将构造函数的作用域赋给新对象(this指向这个新对象)

执行构造函数中的代码

返回新对象

不再派发事件。

终止事件在传播过程的捕获、目标处理或起泡阶段进一步传播。调用该方法后，该节点上处理该事件的处理程序将被调用，事件不再被分派到其他节点。

取消冒泡

var e = (event) ? event : window.event;

if (window.event) {//IE

e.cancelBubble=true;

} else { //别的浏览器

e.stopPropagation();

}

接<a>的默认动作就是跳转到指定页面，阻止它的跳转：

//假定有链接< a href=" " id="testA" >caibaojian.com</ a> var a = document.getElementById("testA"); a.onclick =function(e){ if(e.preventDefault){ e.preventDefault(); }else{ window.event.returnValue == false; } }

javascript的return false只会阻止默认行为，而是用jQuery的话则既阻止默认行为又防止对象冒泡。

为什么要进行模块化开发

1解决命名冲突2.文件依赖

在团队合作中你不会知道自己写的函数或变量等是否会与别人起冲突

改变变量的作用域

可以封装临时变量，避免全局变量的污染。也可以返回一个在全局中需要的变量（用return）

open()方法的第三个参数表示采用异步(true)还是同步(false)处理

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

setTimeout允许讲一个字符串作为第一个参数，而且js内部将会调用eval()函数用来动态执行一段字符串脚本，至于为什么找不到cl函数，我们猜想是作用域问题，既然是eval动态执行，我们在字符串参数中输出当前this绑定的对象：原来this绑定window全局对象，这下明白了，eval()执行动态脚本的时候，在全局作用域并没有找到我们定义在函数test内部的cl，所以会报错。

js 引擎在处理未声明变量时，显然是进行第一种查询。由于作用域中没有未声明的变量，所以查询会一直逐级向上直到全局作用域。这时，非严格模式与严格模式会有两种不同的处理方式。

第一种是 LES 查询。当查询的目的是对变量进行赋值时，比如 「var b = 2」，进行这种查询。第二种查询是 RES 查询。当查询的目的是获取变量的值时，比如 「 if(a)」中的 a，进行这种查询。

非严格模式下，如果作用域中没有所查询变量，全局作用域中就会自动创建一个具有该名称的变量。变量在全局作用域中，所以会变为一个全局变量。而在严格模式下全局作用域中不会自动创建该变量， js 引擎在查询不到变量后就会抛出一个 ReferenceError 异常。

https://www.cnblogs.com/SanMaoSpace/p/3139186.htmljson xml

HTML(HyperText Markup Language)，超文本标记语言。“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素，“标记”是指用特定的标记符号来标记要显示的内容的各个部分。

XML(Extensible Markup Language)，可扩展标记语言。XML是标准通用标记语言的子集，是一种用于标记电子文件使其具有结构性的标记语言，它被设计用来传输和存储数据，是对超文本标记语言的补充。可扩展标记语言是一种元标记语言，即定义了用于定义其他特定领域有关语义的、结构化的标记语言，这些标记语言将文档分成许多部件并对这些部件加以标识。它能够更精确地声明内容，方便跨越多种平台的更有意义的搜索结果。

XHTML(Extensible HyperText Markup Language)，可扩展超文本标记语言。XHTML基于可扩展标记语言（XML）。XHTML就是一个扮演着类似HTML的角色的可扩展标记语言（XML），所以，本质上说，XHTML是一个过渡技术，结合了部分XML的强大功能及大多数HTML的简单特性。

XHTML 是更严谨更纯净的 HTML 版本。它的可扩展性和灵活性将适应未来网络应用更多的需求。