# Google HTML/CSS代码风格指南

修正版本 2.1

## 背景

本文档定义了HTML/CSS的编写格式和风格规则。它旨在提高合作和代码质量,并使其支持基础架构。适用于HTML/CSS文件。

## 样式规则

**目录文件命名**

名称全部用小写英文字母、数字、下划线的组合，其中不得包含汉字、空格和特殊字符。

目录名称应使用意义完整的英文描述或中文描述的首字母

<!-- 推荐 -->

file，bszr

HTML文件名称应使用以下划线分隔的多英文单词或中文描述首字母的形式命名，要求能说明显示内容的信息，命名格式：模块名称\_展示方式

<!-- 推荐 -->

bszrs\_view.html，bszrs\_list.html,bszrs\_change.html

CSS文件名称应以所描述模块的英文名称或中文缩写的小写字母命名。

<!-- 推荐 -->

main.css

### 协议

嵌入式资源书写省略协议头

省略图像、媒体文件、样式表和脚本等URL协议头部声明 ( http: , https: )。如果不是这两个声明的URL则不省略。

省略协议声明，使URL成相对地址，防止内容混淆问题和导致小文件重复下载。

<!-- 不推荐 -->

<script src="http://www.google.com/js/gweb/analytics/autotrack.js"></script>

<!-- 推荐 -->

<script src="//www.google.com/js/gweb/analytics/autotrack.js"></script>

/\* 不推荐 \*/

.example {

background: url(http://www.google.com/images/example);

}

/\* 推荐 \*/

.example {

background: url(//www.google.com/images/example);

}

## 排版规则

### 缩进

每次缩进一个个TAB键。

不要用多个空格来进行缩进或无缩进。

<ul>

<li>Fantastic</li>

<li>Great</li>

</ul>

.example {

color: blue;

}

### 大小写

只用小写字母。

所有的代码都用小写字母：适用于元素名，属性，属性值（除了文本和 CDATA ）， 选择器，特性，特性值（除了字符串）。

<!-- 不推荐 -->

<A HREF="/">Home</A>

<!-- 推荐 -->

<img src="google.png" alt="Google">

### 行尾空格

删除行尾白空格。

行尾空格没必要存在。

<!-- 不推荐 -->

<p>What</p>

<!-- 推荐 -->

<p>Yes please.</p>

## 元数据规则

### 编码

用不带BOM头的 UTF-8编码。

让你的编辑器用没有字节顺序标记的UTF-8编码格式进行编写。

在HTML模板和文件中指定编码 <meta charset="utf-8"> . 不需要制定样式表的编码，它默认为UTF-8.

（更多有关于编码的信息和怎样指定它，请查看 [Character Sets & Encodings in XHTML, HTML and CSS](http://www.w3.org/International/tutorials/tutorial-char-enc/en/all.html)。）

### 注释

尽可能的去解释你写的代码。

用注释来解释代码：它包括什么，它的目的是什么，它能做什么，为什么使用这个解决方案，还是说只是因为偏爱如此呢？

css文件顶部多行注释。

/\*

\*@description:中文说明

\*@author:作者

\*@update:更新时间

\*/

css文件内部单行注释。

/\*注释是哪个模块的样式\*/

HTML内部注释结构。

/!--注释内容--/

（本规则可选，没必要每份代码都描述的很充分，它会增重HTML和CSS的代码。这取决于该项目的复杂程度。在编写代码的时候，为了让它更易懂，注释代码逻辑是挺好的一种习惯，特别是在编写 PHP, Java 和 C#语言时。但是HTML标记语言是非常易懂的，没有必要每行代码都注释下。如果你在自己的HTML文件里看到有非常多注释，那么你就应该反省下了。）

## HTML代码风格规则

### 文档类型

请使用HTML5标准。

HTML5是目前所有HTML文档类型中的首选： <!DOCTYPE html> .

（推荐用HTML文本文档格式，即 text/html . 不要用 XHTML。 XHTML格式，即 [application/xhtml+xml](http://hixie.ch/advocacy/xhtml), 有俩浏览器完全不支持，还比HTML用更多的存储空间。）

### 建议在 html 标签上设置正确的 lang 属性

有助于提高页面的可访问性，如：让语音合成工具确定其所应该采用的发音，令翻译工具确定其翻译语言等。

由于历史原因，有时候不得不继续使用zh-CN。比如中文维基百科，沿用了传统的zh-CN/zh-HK/zh-SG/zh-TW（按照标准应 该使用 zh-cmn-Hans-CN、zh-cmn-Hant-HK、zh-cmn-Hans-SG、zh-cmn-Hant-TW）。这时候，合理的软件行 为，是将 zh-CN 等转化为 zh-cmn-Hans（即转化为最常见的误用所对应的实际标准写法）。[Selectors Level 4](http://dev.w3.org/csswg/selectors/#lang-pseudo)已经加入了对BCP 47高级匹配算法的支持，即支持 :lang(\*-Hans) 的写法。语言的标签表示法的国际标准是 [RFC 4646](http://www.ietf.org/rfc/rfc4646.txt)。

<!-- 推荐 -->

<html lang="zh-CN">

<!-- 简体中文-->

<html lang="zh-cmn-Hans">

<!-- 繁体中文 -->

<html lang="zh-cmn-Hant">

### HTML代码有效性

尽量使用有效的HTML代码。

编写有效的HTML代码，否则很难达到性能上的提升。

用类似这样的工具 [W3C HTML validator](http://validator.w3.org/) 来进行测试。

HTML代码有效性是重要的质量衡量标准，并可确保HTML代码可以正确使用。

<!-- 不推荐 -->

<title>Test</title>

<article>This is only a test.

<!-- 推荐 -->

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Test</title>

</head>

<body>

<article>This is only a test.</article>

</body>

</html>

### 语义

根据HTML各个元素的用途而去使用它们。

使用元素 (有时候错称其为“标签”) 要知道为什么去使用它们和是否正确。 例如，用heading元素构造标题， p 元素构造段落, a 元素构造锚点等。段落之间不要使用<br/> 添加新的一行，而应该使用使用CSS margin 和/或 padding属性即可。对于一列相关的元素，可以使用：

* <ul> (无序列表) 当列表项目的顺序不是很重要时。
* <ol> (有序列表) 当列表项目的顺序非常重要时。
* <dl> (定义列表) 适用于项目/定义对。

不要使用<blockquote>来缩进，只有当需要引用文本的时候才使用它。

根据HTML各个元素的用途而去使用是很重要的，它涉及到文档的可访问性、重用和代码效率等问题。

<!-- 不推荐 -->

<div onclick="goToRecommendations();">All recommendations</div>

<!-- 推荐 -->

<a href="recommendations/">All recommendations</a>

导航是网站设计一个非常重要方面，<ul>元素与CSS的结合会让你的导航菜单更具有意义(因为，毕竟，导航菜单是一个链接列表) ，更加美观。

<!-- 推荐 -->

<ul id="main\_nav">

<li>

<a href="#" class="active">主页</a>

</li>

<li>

<a href="#">About</a>

</li>

</ul>

#main\_nav {

position: absolute;

right: 30px;

top: 40px;

}

#main\_nav li {

float: left;

margin-left: 2px;

}

#main\_nav li a{

font-size: 16px;

color: #fff;

text-decoration: none;

padding: 8px 15px;

-moz-border-radius: 7px;

-webkit-border-radius: 7px;

border-radius: 7px;

}

#main\_nav li a.active,

#main\_nav li a:hover {

background-color: #0b86cb;

}

### 不要什么时候都用divs

有的开发人员经常用<div>标签包装一层又一层的<div>标签，这是非常不可取的。根据[最新的 W3C HTML规范草案](http://dev.w3.org/html5/spec/Overview.html#the-div-element)，<div>是一个毫无意义的元素，应该是"在没有其他任何合适的元素可选择的情况下作为最后一个考虑的元素。"

但是很多程序员就喜欢非常琐碎的东西就用它来表示，如将内联元素显示为块元素。对于Divs的使用一定要谨慎。

### 多媒体后备方案

为多媒体提供备选内容。

对于多媒体，如图像，视频，通过 canvas 读取的动画元素，确保提供备选方案。 对于图像使用有意义的备选文案（ alt ） 对于视频和音频使用有效的副本和文案说明。

提供备选内容是很重要的，原因：给盲人用户以一些提示性的文字，用 @alt 告诉他这图像是关于什么的，给可能没理解视频或音频的内容的用户以提示。

（图像的 alt 属性会产生冗余，如果使用图像只是为了不能立即用CSS而装饰的 ，就不需要用备选文案了，可以写 alt="" 。）

<!-- 不推荐 -->

<img src="spreadsheet.png">

<!-- 推荐 -->

<img src="spreadsheet.png" alt="电子表格截图">

### 关注点分离

将表现和行为分开。

严格保持结构 （标记），表现 （样式），和行为 （脚本）分离, 并尽量让这三者之间的交互保持最低限度。

确保文档和模板只包含HTML结构， 把所有表现都放到样式表里，把所有行为都放到脚本里,既HTML文件里边原则上不直接定义css和js脚本，最好分别放到样式和脚本文件里边。

此外，尽量使脚本和样式表在文档与模板中有最小接触面积，即减少外链。

建议在 head 中引入页面需要的所有 CSS 资源。在页面渲染的过程中，新的CSS可能导致元素的样式重新计算和绘制，页面闪烁。

建议将不是特别重要的JavaScript文件放到文件的底部引入，最好就是放在最后结束的 </body>标签前面

将表现和行为分开维护是很重要滴，因为更改HTML文档结构和模板会比更新样式表和脚本更花费成本。

<!-- 不推荐 -->

<!DOCTYPE html>

<title>HTML sucks</title>

<link rel="stylesheet" href="base.css" media="screen">

<link rel="stylesheet" href="grid.css" media="screen">

<link rel="stylesheet" href="print.css" media="print">

<h1 style="font-size: 1em;">HTML sucks</h1>

<p>I’ve read about this on a few sites but now I’m sure:

<u>HTML is stupid!!1</u>

<center>I can’t believe there’s no way to control the styling of

my website without doing everything all over again!</center>

<!-- 推荐 -->

<!DOCTYPE html>

<html>

<head>

<title>My first CSS-only redesign</title>

<link rel="stylesheet" href="default.css">

</head>

<body>

<h1>My first CSS-only redesign</h1>

<p>I’ve read about this on a few sites but today I’m actually

doing it: separating concerns and avoiding anything in the HTML of

my website that is presentational. </p>

<p>It’s awesome!</p>

</body>

</html>

### 实体引用

不要用实体引用。

不需要使用类似 &mdash; 、 &rdquo; 和 &#x263a; 等的实体引用, 假定团队之间所用的文件和编辑器是同一编码（UTF-8）。

在HTML文档中具有特殊含义的字符（例如 < 和 & )为例外， 噢对了，还有 “不可见” 字符 （例如no-break空格）。

<!-- 不推荐 -->

欧元货币符号是 &ldquo;&eur;&rdquo;。

<!-- 推荐 -->

欧元货币符号是 “€”。

### 结束你的标签

W3C规范中有一条标准就是所有的标签都需要有一个结束标记。有些浏览器可能仍然会正确地渲染没有结束标记的语言，但是不带结束标签的代码在这个标准下是无效的。

<!-- 不推荐 -->

<p>Qed.

<!-- 推荐 -->

<p>Sic.</p>

## HTML代码格式规则

### HTML 标签的使用应该遵循标签的语义

下面是常见标签语义

* p - 段落
* h1,h2,h3,h4,h5,h6 - 层级标题
* strong,em - 强调
* ins - 插入
* del - 删除
* abbr - 缩写
* code - 代码标识
* cite - 引述来源作品的标题
* q - 引用
* blockquote - 一段或长篇引用
* ul - 无序列表
* ol - 有序列表
* dl,dt,dd - 定义列表。

<!-- 不推荐 -->

<div>Esprima serves as an important

<span class="strong">building block</span>

for some JavaScript language tools.

</div>

<!-- 推荐 -->

<p>Esprima serves as an important

<strong>building block</strong>

for some JavaScript language tools.

</p>

### 属性

所有非布尔类型的属性必须有值而且用双引号, 不建议使用单引号或不使用引号,布尔类型的属性，建议不添加属性值。

<!-- 不推荐 -->

<link rel='stylesheet'>

<link rel=stylesheet>

<!-- 推荐 -->

<link rel="stylesheet" checked >

### 标签使用必须符合标签嵌套规则

比如 div 不得置于 p 中，tbody 必须置于 table 中。

详细的标签嵌套规则参见[HTML DTD](http://www.cs.tut.fi/%7Ejkorpela/html5.dtd)中的 Elements 定义部分。

### 元素 id 必须保证页面唯一

同一个页面中，不同的元素包含相同的 id，不符合 id 的属性含义。并且使用 document.getElementById 时可能导致难以追查的问题。

### 同一页面，应避免出现name 与 id的值相同不同元素

IE 浏览器会混淆元素的 id 和 name 属性， document.getElementById 可能获得不期望的元素。所以在对元素的 id 与 name 属性的命名需要非常小心。

一个比较好的实践是，为 id 和 name 使用不同的命名法。

<!-- 不推荐 -->

<input name="foo">

<div id="foo"></div>

<script>

// IE6 将显示 INPUT

alert(document.getElementById('foo').tagName);

</script>

### 格式

每个块元素、列表元素或表格元素都独占一行，每个子元素都相对于父元素进行缩进。

独立元素的样式, 将块元素、列表元素或表格元素都放在新行。

另外，需要缩进块元素、列表元素或表格元素的子元素。

（如果出现了列表项左右空文本节点问题，可以试着将所有的 li 元素都放在一行。)

<blockquote>

<p><em>Space</em>, the final frontier.</p>

</blockquote>

<ul>

<li>Moe</li>

<li>Larry</li>

<li>Curly</li>

</ul>

<table>

<thead>

<tr>

<th scope="col">Income</th>

<th scope="col">Taxes</th>

</tr>

</thead>

<tbody>

<tr>

<td>$ 5.00</td>

<td>$ 4.50</td>

</tr>

</tbody>

</table>

## CSS代码风格规则

### CSS代码有效性

尽量使用有效的CSS代码。

使用有效的CSS代码，除非是处理CSS校验器程序错误或者需要专有语法。

使用有效的CSS是重要的质量衡量标准，如果发现有的CSS代码没有任何效果的可以删除，确保CSS用法适当。

### ID和class的命名

为ID和class取通用且有意义的名字。

应该从ID和class的名字上就能看出这元素是干嘛用的，而不是表象或模糊不清的命名。

应该优先虑以这元素具体目来进行命名，这样也就最容易理解，减少更新。

通用名称可以加在兄弟元素都不特殊或没有个别意义的元素上，可以起名类似“helpers”这样的泛。

使用功能性或通用的名字会减少不必要的文档或模板修改。

/\* 不推荐: 无意义 不易理解 \*/

#yee-1901 {}

/\* 不推荐: 表达不具体 \*/

.button-green {}

.clear {}

/\* 推荐: 明确详细 \*/

#gallery {}

#login {}

.video {}

/\* 推荐: 通用 \*/

.aux {}

.alt {}

class名称中不应该包含颜色（red/blue）、定位(left/right)等于具体显示效果有关的信息。应该用意义命名而不是从class显示效果命名。

/\* 不推荐: 表达不具体 \*/

.red { color:red}

/\* 推荐: 明确详细 \*/

.important-news {

color:red;

}

### ID和class命名风格

非必要的情况下，ID和class的名称应尽量简短。

简要传达ID或class是关于什么的。

通过这种方式，使的代码易懂且高效。

/\* 不推荐 \*/

#navigation {}

.atr {}

/\* 推荐 \*/

#nav {}

.author {}

### 类型选择器

避免使用CSS类型选择器。

非必要的情况下不要使用元素标签名和ID或class进行组合。

出于性能上的考虑避免使用父辈节点做选择器 [performance reasons](http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/).

/\* 不推荐 \*/

ul#example {}

div.error {}

/\* 推荐 \*/

#example {}

.error {}

### 属性缩写

写属性值的时候尽量使用缩写。

CSS很多属性都支持缩写[shorthand](http://www.w3.org/TR/CSS21/about.html#shorthand) （例如 font ） 尽量使用缩写，甚至只设置一个值。

使用缩写可以提高代码的效率和方便理解。

/\* 不推荐 \*/

border-top-style: none;

font-family: palatino, georgia, serif;

font-size: 100%;

line-height: 1.6;

padding-bottom: 2em;

padding-left: 1em;

padding-right: 1em;

padding-top: 0;

/\* 推荐 \*/

border-top: 0;

font: 100%/1.6 palatino, georgia, serif;

padding: 0 1em 2em;

### 0和单位

省略0后面的单位。

非必要的情况下 0 后面不用加单位。如果没有边框时，不要写成border:0;,应写成border:none;

margin: 0;

padding: 0;

border:none;

### URI外的引号

省略URI外的引号。

不要在 url() 里用 ( "" , '' ) 。

@import url(//www.google.com/css/go.css);

### 前缀

选择器前面加上特殊应用标识的前缀（可选）。

大型项目中最好在ID或class名字前加上这种标识性前缀（命名空间），使用短破折号链接。

使用命名空间可以防止命名冲突，方便维护，比如在搜索和替换操作上。

.adw-help {} /\* AdWords \*/

#maia-note {} /\* Maia \*/

### ID和class命名的定界符

ID和class名字有多单词组合的用短破折号“-”分开。

别在选择器名字里用短破折号“-”以外的连接词(包括啥也没有)， 以增进对名字的理解和查找。

/\* 不推荐：“demo”和“image”中间没加“-” \*/

.demoimage {}

/\* 不推荐：用下划线“\_”是屌丝的风格 \*/

.error\_status {}

/\* 推荐 \*/

#video-id {}

.ads-sample {}

## CSS代码格式规则

### 声明顺序

依字母顺序进行声明。

都按字母顺序声明，很容易记住和维护。

忽略浏览器的特定前缀排序，但多浏览器特定的某个CSS属性前缀应相对保持排序（例如-moz前缀在-webkit前面）。

background: fuchsia;

border: 1px solid;

-moz-border-radius: 4px;

-webkit-border-radius: 4px;

border-radius: 4px;

color: black;

text-align: center;

text-indent: 2em;

### 代码块内容缩进

缩进所有代码块（“{}”之间）内容。

缩进所有[代码块](http://www.w3.org/TR/CSS21/syndata.html#block)的内容，它能够提高层次结构的清晰度。

@media screen, projection {

html {

background: #fff;

color: #444;

}

}

### 声明完结

所有声明都要用“;”结尾，";"后不要空格。

考虑到一致性和拓展性，请在每个声明尾部都加上分号。

/\* 不推荐 \*/

.test {

display: block;

height: 100px

}

/\* 推荐 \*/

.test {

display: block;

height: 100px;

}

### 属性名完结

在属性名冒号结束后加一个空字符。

出于一致性的原因，在属性名和值之间加一个空格（可不是属性名和冒号之间噢），在{}大括号开始前六空格，结束后不要空格。

/\* 不推荐 \*/

h3{

font-weight:bold;

}

/\* 推荐 \*/

h3 {

font-weight: bold;

}

### 选择器和声明分行

将选择器和声明隔行。

每个选择器和声明都要独立新行。

/\* 不推荐 \*/

a:focus, a:active {

position: relative; top: 1px;

}

/\* 推荐 \*/

h1,

h2,

h3 {

font-weight: normal;

line-height: 1.2;

}

### 规则分行

每个规则独立一行。

两个规则之间隔行。

html {

background: #fff;

}

body {

margin: auto;

width: 50%;

}