## **样式规则**

### **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**

每次缩进两个空格。

不要用TAB键或多个空格来进行缩进。

<ul>

<li>Fantastic

<li>Great

</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?\_

<!-- 推荐 -->

<p>Yes please.

## **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？

（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。）

### **活动的条目**

用 TODO 标记代办事项和正活动的条目

只用 TODO 来强调代办事项， 不要用其他的常见格式，例如 @@ 。

附加联系人（用户名或电子邮件列表），用括号括起来，例如 TODO(contact) 。

可在冒号之后附加活动条目说明等，例如 TODO: 活动条目说明 。

{# TODO(cha.jn): 重新置中 #}

<center>Test</center>

<!-- TODO: 删除可选元素 -->

<ul>

<li>Apples</li>

<li>Oranges</li>

</ul>

## **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。）

### **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>

<meta charset="utf-8">

<title>Test</title>

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

### **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。

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>

### **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。

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。

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>

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

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

<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>It’s awesome!

### **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 “€”。

### **可选标签**

省略可选标签（可选）。

出于优化文件大小和校验， 可以考虑省略可选标签，哪些是可选标签可以参考 [HTML5 specification](http://www.whatwg.org/specs/web-apps/current-work/multipage/syntax.html" \l "syntax-tag-omission)。

（这种方法可能需要更精准的规范来制定，众多的开发者对此的观点也都不同。考虑到一致性和简洁的原因，省略所有可选标记是有必要的。）

<!-- 不推荐 -->

<!DOCTYPE html>

<html>

<head>

<title>Spending money, spending bytes</title>

</head>

<body>

<p>Sic.</p>

</body>

</html>

<!-- 推荐 -->

<!DOCTYPE html>

<title>Saving money, saving bytes</title>

<p>Qed.

### **type属性**

在样式表和脚本的标签中忽略 type 属性

在样式表（除非不用 CSS）和脚本（除非不用 JavaScript）的标签中 不写 type 属性。

HTML5默认 type 为[text/css](http://www.whatwg.org/specs/web-apps/current-work/multipage/semantics.html" \l "attr-style-type)和[text/javascript](http://www.whatwg.org/specs/web-apps/current-work/multipage/scripting-1.html" \l "attr-script-type)类型，所以没必要指定。即便是老浏览器也是支持的。

<!-- 不推荐 -->

<link rel="stylesheet" href="//www.google.com/css/maia.css"

type="text/css">

<!-- 推荐 -->

<link rel="stylesheet" href="//www.google.com/css/maia.css">

<!-- 不推荐 -->

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

type="text/javascript"></script>

<!-- 推荐 -->

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

## **HTML代码格式规则**

### **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（as CSS allows elements to assume a different role per display property), 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。 A linter is encouraged to throw a warning instead of an error.)

<blockquote>

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

</blockquote>

<ul>

<li>Moe

<li>Larry

<li>Curly

</ul>

<table>

<thead>

<tr>

<th scope="col">Income

<th scope="col">Taxes

<tbody>

<tr>

<td>$ 5.00

<td>$ 4.50

</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。

用类似[W3C CSS validator](http://jigsaw.w3.org/css-validator/) 这样的工具来进行有效性的测试。

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，这样他就最容易理解，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 {}

### **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，似的代码易懂且高效。

/\* 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" \l "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。

margin: 0;

padding: 0;

### **0开头的小数**

省略0开头小数点前面的0。

值或长度在-1与1之间的小数，小数前的 0 可以忽略不写。

font-size: .8em;

### **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);

### **十六进制**

十六进制尽可能使用3个字符。

加颜色值时候会用到它，使用3个字符的十六进制更短与简洁。

/\* 不推荐 \*/

color: #eebbcc;

/\* 推荐 \*/

color: #ebc;

### **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 {}

### **Hacks**

最好避免使用该死的CSS “hacks” —— 请先尝试使用其他的解决方法。

虽然它很有诱惑力，可以当作用户代理检测或特殊的CSS过滤器，但它的行为太过于频繁，会长期伤害项目的效率和代码管理，所以能用其他的解决方案就找其他的。

## **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" \l "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。

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）。

/\* 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%;

}

## **CSS元数据规则**

### **注释部分**

按组写注释。（可选）

如果可以，按照功能的类别来对一组样式表写统一注释。独立成行。

/\* Header \*/

#adw-header {}

/\* Footer \*/

#adw-footer {}

/\* Gallery \*/

.adw-gallery {}