**前端HTML+CSS规范**

1. **摘要**

收集互联网公司对前端 HTML 、CSS 、 Javascript 等一些规范，结合自身当前实际。总结如下，部分删减。希望大家在书写规范上起到一定作用。

**为何规范？**

* 规范不是一种约束，而是一种约定，强调团队的一致性
* 加强团队的合作性，提高协作效率
* 形成一种团队文件，积累知识
* 规范最终是为项目服务的，我们所做的一切都是为了优化项目和流程提高我们的工作效率
* 规范的好处是，时间长了，看代码你就能区分出什么是“专业”。客户，同行可以透过严谨的代码书写看到公司背后的严谨和专业

**基本准则**

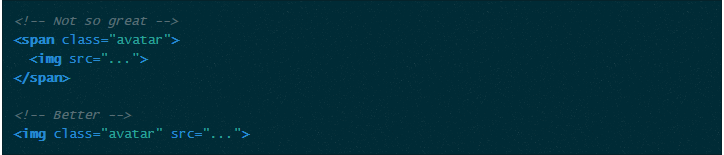
* 一个项目应该永远遵循同一套编码规范！
* 不管有多少人共同参与同一项目，一定要确保每一行代码都像是同一个人编写的。
* 符合web标准，语义化html，结构表现行为分离，兼容性优良，页面性能方面，代码要求简洁明了有序，任何时候都要尽量使用最少的标签并保持最小的复杂度，尽可能的减小服务器负载，保证最快的解析速度。总的来说就是：代码简、易维护、性能高

**开发及测试工具约定**

* 开发过程中严格按分工完成页面，以提高css复用率，避免重复开发。
* 书写所有人都可以看的懂的代码，简洁易懂是一种美德，为用户着想，为服务器着想。
* 编码必须格式化，比如缩进。至少看起来都美观及舒服。
* 主兼容浏览器：Firefox、IE7-IE11、Chrome、360、Safari。次兼容浏览器Opera、QQ、搜狗等。

1. **HTML书写规范**
2. 所有编码均遵循xhtml标准，标签属性命名必须用小写字母，且所有标签必须闭合，包括br(<br/>),img,link等，属性必须用双引号包括；（注：有些文档说img这类属于自闭标签，不需要加斜线结束）



1. 充分利用无兼容性问题的html自身标签，比如span,em,strong,label等等；需要为html元素添加自定义属性的时候，首先要考虑下有没有默认的已有的合适标签去设置，如果没有，可以使用须以”data-”为前缀来添加自定义属性，避免使用”data:”等其他命名方式；
2. 语义化html，如标题根据重要性用h\*(同一页面只能有一个h1),段落标记用p,列表用ul,内联元素中不可嵌套块级元素；
3. 编写 HTML 代码时，尽可能减少div嵌套及避免多余的父元素。比如<div><h2>　</h2></div>已经满足要求，那么就不要再写成<div><div><h2></h2></div></div。很多时候，这需要迭代和重构来实现。请看下面的案例：
4. 在页面中尽量避免使用style属性，即style=”…”。
5. 严禁在文档中使用<style type="text/css"></style>代码块。
6. 能以背影形式呈现的图片，尽量写入css样式中。
7. 一些浏览器会将含有这些词的作为广告拦截，文件命名、ID、CLASS等所有命名避免以下词汇。`ad`、`ads`、`adv`、`banner`、`sponsor`、`gg`、`guangg`、`guanggao`等。
8. 特殊符号应使用转义符。如：<（&lt;）、>（&gt;）、¥（&yen;）、©（&copy;）等。
9. HTML中对于属性的定义，确保全部使用双引号，绝不要使用单引号。
10. 验证代码，推荐w3c验证地址：<http://validator.w3.org/#validate_by_input>

人不免会出错，我们编写代码的时候也是一样的，你有时候总会小写或多写，比如说忘了关闭你的元素标签，不记得写上元素必须的属性，虽然有一些错误不会给你带来什么灾难性的后果，但也不免会给你带来你无法意估的错误。所以建议您写完代码的时候去验证你一下你的代码。验证后的代码总是比不验证的代码强。如果你的代码没有任何错误，W3C验证工具会在你们面前呈现绿色的文字，这样让你是无比的激动人心，因为再次证明了你写的代码经得起W3C的标准。

1. 一个标签上引用的Class Name不要过多，越少越好。比如不要出现这种情况：<div class="class1 class2 class3 class4"></div>。
2. 对于一个语义化的内部标签，应尽量避免使用Class Name。比如在这样一个列表中，li标签中的itm应去除：<ul class="m-help"><li class="itm"></li><li class="itm"></li></ul>。
3. 书写链接地址时, 必须避免重定向，例如：href=”http://abc.com/”, 即须在URL地址后面加上“/”。
4. 合理的避免使用ID。一般情况下id不应该被应用于样式，应尽可能留给后台程序开发人员使用。
5. **CSS书写规范**

**CSS选择器匹配机制**

如子选择器规则　#header > a{…}

我们中的大多数人都是从左到右的阅读习惯，会习惯性的设定浏览器也是从左到右的方式进行匹配规则，推测这条规则的开销并不高。

我们会假设浏览器以这样的方式工作：寻找 id 为 header 的元素，然后将样式规则应用到直系子元素中的 a 元素上。我们知道文档中只有一个 id 为 header 的元素，并且它只有几个 a 元素的子节点，所以这个CSS选择器应该相当高效。

事实上，却恰恰相反，CSS选择器是从右到左进行规则匹配。了解这个机制后，例子中看似高效的选择器在实际中的匹配开销是很高的，浏览器必须遍历页面中所有的 a 元素并且确定其父元素的 id 是否为 header 。

Google 资深web开发工程师 Steve Souders 对 CSS 选择器的执行效率从高到低做了一个排序：

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）

根据以上「选择器匹配」与「选择器执行效率」原则，我们可以通过避免不恰当的使用，提升 CSS 选择器性能。

**命名**

1. class 名称中只能出现小写字母数字和破折号（dashe）（不是下划线，也不是驼峰命名法）。破折号应当用于相关 class 的命名（类似于命名空间）（例如，.btn 和 .btn-danger）。
2. 避免过度任意的简写。.btn 代表 button，但是 .s 不能表达任何意思。

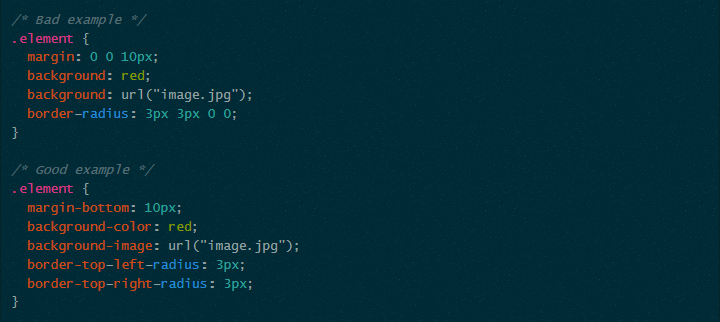


1. class 名称应当尽可能短，并且意义明确。
2. 使用有意义的名称。使用有组织的或目的明确的名称，不要使用表现形式（presentational）的名称。

**简写声明**

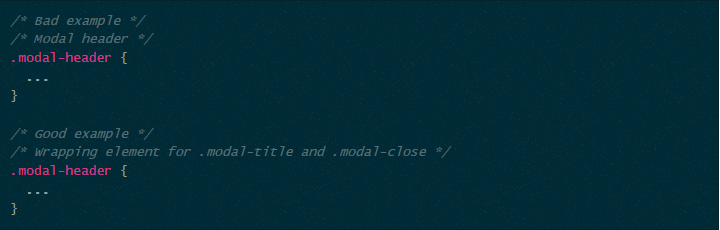
在需要显示地设置所有值的情况下，应当尽量限制使用简写形式的属性声明。常见的滥用简写属性声明的情况如下：padding、margin、background、border、border- radius

大部分情况下，我们不需要为简写形式的属性声明指定所有值。例如，HTML 的 heading 元素只需要设置上、下边距（margin）的值，因此，在必要的时候，只需覆盖这两个值就可以。过度使用简写形式的属性声明会导致代码混乱，并且会对属性值带来不必要的覆盖从而引起意外的副作用。



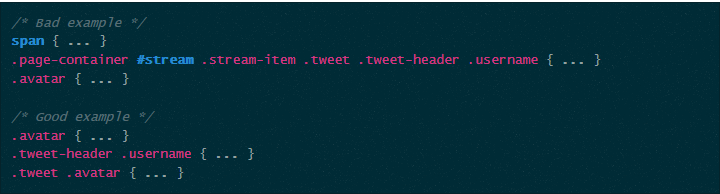
**注释**

* 必须为大区块样式添加注释，小区块适量注释，团队开发这个非常重要，尽管很多时候大家不愿意遵守，但确实重要！！！
* 对于较长的注释，务必书写完整的句子；对于一般性注解，可以书写简洁的短语。



**选择器**

* 对于经常出现的组件，避免使用属性选择器（例如，[class^="..."]）。浏览器的性能会受到这些因素的影响。
* 选择器要尽可能短，并且尽量限制组成选择器的元素个数，建议不要超过 3 。
* **只有**在必要的时候才将 class 限制在最近的父元素内（也就是后代选择器）（例如，不使用带前缀的 class 时 – 前缀类似于命名空间）。



**声明顺序**

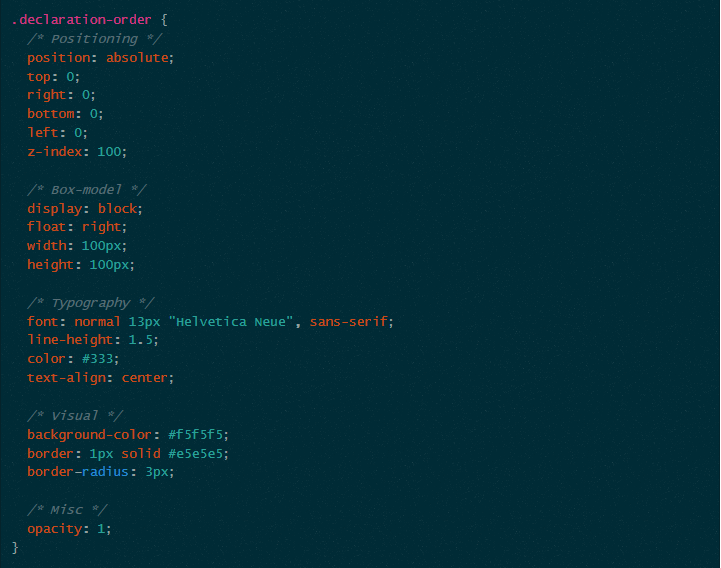
相关的属性声明应当归为一组，并按照下面的顺序排列：

* Positioning
* Box model
* Typographic（文本相关）
* Visual（视觉效果）

由于定位（positioning）可以从正常的文档流中移除元素，并且还能覆盖盒模型（box model）相关的样式，因此排在首位。盒模型排在第二位，因为它决定了组件的尺寸和位置。

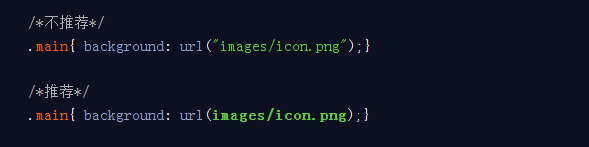
其他属性只是影响组件的内部（inside）或者是不影响前两组属性，因此排在后面。

简单举个例子，网页中的图片，如果没有设置width和height，在图片载入之前，他所占的空间为0，但是当他加载完毕之后，那块为0的空间突然被撑开了，这样会导致，他下面的元素重新排列和渲染，造成重绘（repaint）和回流（reflow）。我们在写css的时候，把元素的定位放在前头，首先让浏览器知道该元素是在文本流内还是外，具体在页面的哪个部位，接着让浏览器知道他们的宽度和高度，border等这些占用空间的属性，其他的属性都是在这个固定的区域内渲染的，差不多就是这个意思吧。



**其他**

1. 非特殊情况下样式文件必须外链至<head>…</head>之间。
2. 避免为 0 值指定单位，例如，用 margin: 0; 代替 margin: 0px;。
3. 对于属性值或颜色参数，省略小于 1 的小数前面的 0 （例如，.5 代替 0.5；-.5px 代替 -0.5px）。
4. 尽量使用简写形式的十六进制值并且颜色值小写，例如，用 #fff 代替 #ffffff。
5. 通过 Sass 或 Less 类似的 CSS 预处理器将多个 CSS 文件编译为一个文件。
6. 请使用W3C的CSS验证器进行验证。如果你坚持认为你的布局没有什么可以修改的时候，你可以使用CSS验证器对找出错误会有很大帮助。http://jigsaw.w3.org/css-validator/#validate\_by\_input
7. 尽量优化使样式轻量化，因为样式越小越容易加载，越小越容易维护及更新。
8. 链接的样式，务必按照这个顺序来书写a:link -> a:visited -> a:hover -> a:active。
9. 书写css3代码时，务必要考虑不支持浏览器的展示效果。
10. 避免在css中使用 @import。
11. 字体粗细采用具体数值，粗体bold写为700，正常normal写为400。
12. 尽量不要在CSS中使用!important。
13. 绝对不要在CSS中使用”\*”选择符。
14. 每个声明结束都应该带一个分号，不管是不是最后一个声明。
15. 如果没有边框时，不要写成border:0，应该写成border:none 。
16. 避免在CSS中使用expression。
17. 使用z-index属性尽量z-index的值不要超过150（通用组的除外），页面中的元素内容的z-index不能超过10（提示框等模块除外但维持在150以下），不允许直接使用（999~9999）之间大值，禁止层级间盲目攀比。
18. 避免使用长选择符，如.header .nav ul li a{…}；或者是子选择符，如.header > .nav > ul > li > a{…}。
19. 图片URL值不要使用引号。

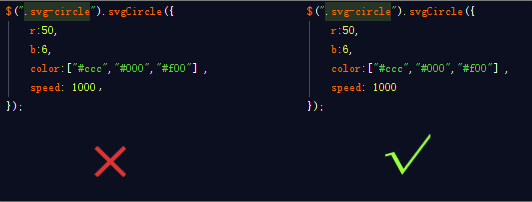


1. 不需要重复定义可继承的值：子元素自动继承父元素的属性值，像颜色、字体等已经在父元素中定义过的，在子元素中可以直接继承，不需要重复定义。但是要注意，浏览器可能用一些默认的值覆盖你的定义。
2. 如果能通过合理的 HTML 结构或使用其他的 CSS 定义达到理想的样式，则不应该使用 hack 手段解决问题。通常 hack 会导致维护成本的增加。
3. **图片规范**

* 背景图片请尽可能使用sprite技术, 减小http请求, 考虑到多人协作开发, sprite按照模块、业务、页面来划分均可。
* 图片格式仅限于gif、png、jpg，除需要透明的图片使用png、gif外其他图片都采用jpg格式。为尽量控制文件大小，大尺寸图片应避免使用png，尽量使用jpg。
* 重要图片必须加上alt属性；给重要的元素和截断的元素加上title。
* 在保证视觉效果的情况下选择最小的图片格式与图片质量，以减少加载时间。

1. **JAVASCRIPT书写规范**

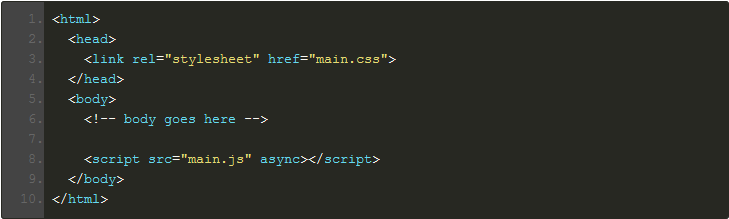
* 如果css可以做到，就不要使用js。
* 书写过程中，每行代码结束必须要有分号。
* 为代码增加适量注释。
* 避免额外的逗号，如下图：



* 通过 JavaScript 生成的标签让内容变得不易查找、编辑，并且降低性能。能避免时尽量避免。
* 每个页面只使用一次document的ready事件，这样便于调试与行为流跟踪。
* 变量、函数在使用前必须先定义，原则上应避免全局变量。
* 每个 var 只能声明一个变量，一个 var 声明多个变量，容易导致较长的行长度，并且在修改时容易造成逗号和分号的混淆。



* 变量必须 即用即声明，不得在函数或其它形式的代码块起始位置统一声明所有变量。变量声明与使用的距离越远，出现的跨度越大，代码的阅读与维护成本越高。虽然JavaScript的变量是函数作用域，还是应该根据编程中的意图，缩小变量出现的距离空间。
* 非特殊情况下JavaScript文件必须外链至页面底部，如



* 其它更多规范推荐阅读https://github.com/ecomfe/spec/blob/master/javascript-style-guide.md

1. **页面模块结构常用命名**

|  |  |  |  |
| --- | --- | --- | --- |
| 样式名称 | 命名 | 样式名称 | 命名 |
| 头部 | header | 尾部 | footer |
| 侧栏 | sidebar | 栏目 | column |
| 导航 | nav | 主导航 | main-nav |
| 子导航 | sub-nav | 顶部导航 | top-nav |
| 菜单 | menu | 子菜单 | sub-menu |
| 模块(如：新闻模块) | mod (mod-news) | 面包屑 | breadcrumb |
| 内容 | content | 次级内容 | sub-content |
| 标志 | logo | 滚动 | scroll |
| 标签页 | tab | 列表 | list |
| 当前 | current | 提示小技巧 | tips |
| 图标 | icon | 版权 | copyright |
| 按钮 | btn | Banner图 | slider |
| 状态 | status |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **结语**

* 规范只是约定的一种形式，没有绝对的对与错。所选观点仅代表本人认可的观点。
* 由于来源不同文档，文中的严禁，禁止，尽量，绝对等等在此文档都可以理解为同一个意思，即尽量不要。
* 推荐大家阅读Web开发规范文档：<http://www.kancloud.cn/chandler/css-code-guide/50867>