

CSS编码规范

1 代码风格

1.1 缩进

[强制] 使用 4 个空格做为一个缩进层级，不允许使用 2 个空格 或 tab 字符。

示例：

```
.selector {  
    margin: 0;  
    padding: 0;  
}
```

1.2 空格

[强制] 选择器 与 { 之间必须包含空格。

示例：

```
.selector {  
}
```

[强制] 属性名 与之后的：之间不允许包含空格，：与 属性值 之间必须包含空格。

示例：

```
margin: 0;
```

[强制] 列表型属性值 书写在单行时，，后必须跟一个空格。

示例：

```
font-family: Arial, sans-serif;
```

1.3 行长度

[建议] 每行不得超过 120 个字符，除非单行不可分割。

解释：

常见不可分割的场景为URL超长。

[建议] 对于超长的样式，在样式值的 空格 处或 , 后换行，建议按逻辑分组。

示例：

```
/* 不同属性值按逻辑分组 */
background:
    transparent url(aVeryVeryVeryLongUrlIsPlacedHere)
    no-repeat 0 0;

/* 可重复多次的属性，每次重复一行 */
background-image:
    url(aVeryVeryVeryLongUrlIsPlacedHere)
    url(anotherVeryVeryVeryLongUrlIsPlacedHere);

/* 类似函数的属性值可以根据函数调用的缩进进行 */
background-image: -webkit-gradient(
    linear,
    left bottom,
    left top,
    color-stop(0.04, rgb(88,94,124)),
    color-stop(0.52, rgb(115,123,162))
);
```

1.4 选择器

[强制] 当一个 rule 包含多个 selector 时，每个选择器声明必须独占一行。

示例：

```
/* good */
.post,
```

```
.page,  
.comment {  
    line-height: 1.5;  
}  
  
/* bad */  
.post, .page, .comment {  
    line-height: 1.5;  
}
```

[强制] `>`、`+`、`~` 选择器的两边各保留一个空格。

示例：

```
/* good */  
main > nav {  
    padding: 10px;  
}  
  
label + input {  
    margin-left: 5px;  
}  
  
input:checked ~ button {  
    background-color: #69C;  
}  
  
/* bad */  
main>nav {  
    padding: 10px;  
}  
  
label+input {  
    margin-left: 5px;  
}  
  
input:checked~button {
```

```
background-color: #69C;  
}
```

[强制] 属性选择器中的值必须用双引号包围。

解释：

不允许使用单引号，不允许不使用引号。

示例：

```
/* good */  
article[character="juliet"] {  
    voice-family: "Vivien Leigh", victoria, female  
}  
  
/* bad */  
article[character='juliet'] {  
    voice-family: "Vivien Leigh", victoria, female  
}
```

1.5 属性

[强制] 属性定义必须另起一行。

示例：

```
/* good */  
.selector {  
    margin: 0;  
    padding: 0;  
}  
  
/* bad */  
.selector { margin: 0; padding: 0; }
```

[强制] 属性定义后必须以分号结尾。

示例：

```
/* good */  
.selector {  
    margin: 0;  
}  
  
/* bad */  
.selector {  
    margin: 0  
}
```

2 通用

2.1 选择器

[强制] 如无必要，不得为 **id**、**class** 选择器添加类型选择器进行限定。

解释：

在性能和维护性上，都有一定的影响。

示例：

```
/* good */  
#error,  
.danger-message {  
    font-color: #c00;  
}  
  
/* bad */  
dialog#error,  
p.danger-message {  
    font-color: #c00;  
}
```

[建议] 选择器的嵌套层级应不大于 3 级，位置靠后的限定条件应尽可能精确。

示例：

```
/* good */  
#username input {}  
.comment .avatar {}  
  
/* bad */  
.page .header .login #username input {}  
.comment div * {}
```

2.2 属性缩写

[建议] 在可以使用缩写的情况下，尽量使用属性缩写。

示例：

```
/* good */  
.post {  
  font: 12px/1.5 arial, sans-serif;  
}  
  
/* bad */  
.post {  
  font-family: arial, sans-serif;  
  font-size: 12px;  
  line-height: 1.5;  
}
```

[建议] 使用 **border** / **margin** / **padding** 等缩写时，应注意隐含值对实际数值的影响，确实需要设置多个方向的值时才使用缩写。

解释：

border / **margin** / **padding** 等缩写会同时设置多个属性的值，容易覆盖不需要覆盖的设定。如某些方向需要继承其他声明的值，则应该分开设置。

示例：

```
/* centering <article class="page"> horizontally and highlight featured ones */
```

```
article {  
    margin: 5px;  
    border: 1px solid #999;  
}
```

```
/* good */
```

```
.page {  
    margin-right: auto;  
    margin-left: auto;  
}
```

```
.featured {  
    border-color: #69c;  
}
```

```
/* bad */
```

```
.page {  
    margin: 5px auto; /* introducing redundancy */  
}
```

```
.featured {  
    border: 1px solid #69c; /* introducing redundancy */  
}
```

2.3 属性书写顺序

[建议] 同一 **rule set** 下的属性在书写时，应按功能进行分组，并以 **Formatting Model**（布局方式、位置）> **Box Model**（尺寸）> **Typographic**（文本相关）> **Visual**（视觉效果）的顺序书写，以提高代码的可读性。

解释：

- Formatting Model 相关属性包括：position / top / right / bottom / left / float / display / overflow 等
- Box Model 相关属性包括：border / margin / padding / width / height 等
- Typographic 相关属性包括：font / line-height / text-align / word-wrap 等
- Visual 相关属性包括：background / color / transition / list-style 等

另外，如果包含 **content** 属性，应放在最前面。

示例：

```
.sidebar {  
  /* formatting model: positioning schemes / offsets / z-indexes / display / ... */  
  position: absolute;  
  top: 50px;  
  left: 0;  
  overflow-x: hidden;  
  
  /* box model: sizes / margins / paddings / borders / ... */  
  width: 200px;  
  padding: 5px;  
  border: 1px solid #ddd;  
  
  /* typographic: font / aligns / text styles / ... */  
  font-size: 14px;  
  line-height: 20px;  
  
  /* visual: colors / shadows / gradients / ... */  
  background: #f5f5f5;  
  color: #333;  
  -webkit-transition: color 1s;  
  -moz-transition: color 1s;  
  transition: color 1s;  
}
```

2.4 清除浮动

[建议] 当元素需要撑起高度以包含内部的浮动元素时，通过对伪类设置 **clear** 或触发 **BFC** 的方式进行 **clearfix**。尽量不使用增加空标签的方式。

解释：

触发 BFC 的方式很多，常见的有：

- float 非 none

- position 非 static
- overflow 非 visible

如希望使用更小副作用的清除浮动方法，参见 [A new micro clearfix hack](#) 一文。

另需注意，对已经触发 BFC 的元素不需要再进行 clearfix。

2.5 !important

[建议] 尽量不使用 **!important** 声明。

[建议] 当需要强制指定样式且不允许任何场景覆盖时，通过标签内联和 **!important** 定义样式。

解释：

必须注意的是，仅在设计上 **确实不允许任何其它场景覆盖样式** 时，才使用内联的 **!important** 样式。通常在第三方环境的应用中使用这种方案。下面的 z-index 章节是其中一个特殊场景的典型样例。

2.6 z-index

[建议] 将 **z-index** 进行分层，对文档流外绝对定位元素的视觉层级关系进行管理。

解释：

同层的多个元素，如多个由用户输入触发的 Dialog，在该层级内使用相同的 **z-index** 或递增 **z-index**。

建议每层包含100个 **z-index** 来容纳足够的元素，如果每层元素较多，可以调整这个数值。

[建议] 在可控环境下，期望显示在最上层的元素，**z-index** 指定为 **999999**。

解释：

可控环境分成两种，一种是自身产品线环境；还有一种是可能会被其他产品线引用，但是不会被外部第三方的产品引用。

不建议取值为 **2147483647**。以便于自身产品线被其他产品线引用时，当遇到层级覆盖冲突的情况，留出向上调整的空间。

[建议] 在第三方环境下，期望显示在最上层的元素，通过标签内联和 **!important**，将 **z-index** 指定为 **2147483647**。

解释：

第三方环境对于开发者来说完全不可控。在第三方环境下的元素，为了保证元素不被其页面其他样式定义覆盖，需要采用此做法。

3 值与单位

3.1 文本

[强制] 文本内容必须用双引号包围。

解释：

文本类型的内容可能在选择器、属性值等内容中。

示例：

```
/* good */  
html[lang|"zh"] q:before {  
    font-family: "Microsoft YaHei", sans-serif;  
    content: "";  
}  
  
html[lang|"zh"] q:after {  
    font-family: "Microsoft YaHei", sans-serif;  
    content: "";  
}  
  
/* bad */  
html[lang|=zh] q:before {  
    font-family: 'Microsoft YaHei', sans-serif;  
    content: '';  
}  
  
html[lang|=zh] q:after {  
    font-family: "Microsoft YaHei", sans-serif;  
    content: "";  
}
```

3.2 数值

[强制] 当数值为 **0 - 1** 之间的小数时，省略整数部分的 **0**。

示例：

```
/* good */  
panel {  
  opacity: .8  
}  
  
/* bad */  
panel {  
  opacity: 0.8  
}
```

3.3 url()

[强制] **url()** 函数中的路径不加引号。

示例：

```
body {  
  background: url(bg.png);  
}
```

[建议] **url()** 函数中的绝对路径可省去协议名。

示例：

```
body {  
  background: url(//baidu.com/img/bg.png) no-repeat 0 0;  
}
```

3.4 长度

[强制] 长度为 **0** 时须省略单位。（也只有长度单位可省）

示例：

```
/* good */  
body {  
    padding: 0 5px;  
}  
  
/* bad */  
body {  
    padding: 0px 5px;  
}
```

3.5 颜色

[强制] RGB颜色值必须使用十六进制记号形式 #rrggbb。不允许使用 rgb()。

解释：

带有alpha的颜色信息可以使用 rgba()。使用 rgba() 时每个逗号后必须保留一个空格。

示例：

```
/* good */  
.success {  
    box-shadow: 0 0 2px rgba(0, 128, 0, .3);  
    border-color: #008000;  
}  
  
/* bad */  
.success {  
    box-shadow: 0 0 2px rgba(0,128,0,.3);  
    border-color: rgb(0, 128, 0);  
}
```

[强制] 颜色值可以缩写时，必须使用缩写形式。

示例：

```
/* good */  
.success {  
    background-color: #aca;  
}  
  
/* bad */  
.success {  
    background-color: #aaccaa;  
}
```

[强制] 颜色值不允许使用命名色值。

示例：

```
/* good */  
.success {  
    color: #90ee90;  
}  
  
/* bad */  
.success {  
    color: lightgreen;  
}
```

[建议] 颜色值中的英文字符采用小写。如不用小写也需要保证同一项目内保持大小写一致。

示例：

```
/* good */  
.success {  
    background-color: #aca;  
    color: #90ee90;  
}  
  
/* good */  
.success {  
    background-color: #ACA;
```

```
color: #90EE90;
}

/* bad */
.success {
  background-color: #ACA;
  color: #90ee90;
}
```

3.6 2D 位置

[强制] 必须同时给出水平和垂直方向的位置。

解释：

2D 位置初始值为 **0% 0%**，但在只有一个方向的值时，另一个方向的值会被解析为 center。为避免理解上的困扰，应同时给出两个方向的值。[background-position](#)属性值的定义

示例：

```
/* good */
body {
  background-position: center top; /* 50% 0% */
}

/* bad */
body {
  background-position: top; /* 50% 0% */
}
```

4 文本编排

4.1 字体族

[强制] **font-family** 属性中的字体族名称应使用字体的英文 **Family Name**，其中如有空格，须放置在引号中。

解释：

所谓英文 Family Name，为字体文件的一个元数据，常见名称如下：

字体	操作系统	Family Name
宋体 (中易宋体)	Windows	SimSun
黑体 (中易黑体)	Windows	SimHei
微软雅黑	Windows	Microsoft YaHei
微软正黑	Windows	Microsoft JhengHei
华文黑体	Mac/iOS	STHeiti
冬青黑体	Mac/iOS	Hiragino Sans GB
文泉驿正黑	Linux	WenQuanYi Zen Hei
文泉驿微米黑	Linux	WenQuanYi Micro Hei

示例：

```
h1 {  
    font-family: "Microsoft YaHei";  
}
```

[强制] **font-family** 按「西文字体在前、中文字体在后」、「效果佳 (质量高/更能满足需求) 的字体在前、效果一般的字体在后」的顺序编写，最后必须指定一个通用字体族(**serif / sans-serif**)。

解释：

更详细说明可参考[本文](#)。

示例：

```
/* Display according to platform */  
.article {  
    font-family: Arial, sans-serif;  
}  
  
/* Specific for most platforms */  
h1 {
```

```
font-family: "Helvetica Neue", Arial, "Hiragino Sans GB", "WenQuanYi Micro Hei", "Microsoft YaHei", sans-serif;
}
```

[强制] **font-family** 不区分大小写，但在同一个项目中，同样的 **Family Name** 大小写必须统一。

示例：

```
/* good */
body {
    font-family: Arial, sans-serif;
}

h1 {
    font-family: Arial, "Microsoft YaHei", sans-serif;
}

/* bad */
body {
    font-family: arial, sans-serif;
}

h1 {
    font-family: Arial, "Microsoft YaHei", sans-serif;
}
```

4.2 字号

[强制] 需要在 **Windows** 平台显示的中文内容，其字号应不小于 **12px**。

解释：

由于 Windows 的字体渲染机制，小于 12px 的文字显示效果极差、难以辨认。

4.3 字体风格

[建议] 需要在 **Windows** 平台显示的中文内容，不要使用除 **normal** 外的 **font-style**。其他平台也应慎用。

解释：

由于中文字体没有 italic 风格的实现，所有浏览器下都会 fallback 到 oblique 实现 (自动拟合为斜体)，小字号下 (特别是 Windows 下会在小字号下使用点阵字体的情况下) 显示效果差，造成阅读困难。

4.4 字重

[强制] font-weight 属性必须使用数值方式描述。

解释：

CSS 的字重分 100 – 900 共九档，但目前受字体本身质量和浏览器的限制，实际上支持 400 和 700 两档，分别等价于关键词 normal 和 bold。

浏览器本身使用一系列[启发式规则](#)来进行匹配，在 <700 时一般匹配字体的 Regular 字重，>=700 时匹配 Bold 字重。

但已有浏览器开始支持 =600 时匹配 Semibold 字重 (见[此表](#))，故使用数值描述增加了灵活性，也更简短。

示例：

```
/* good */
h1 {
  font-weight: 700;
}

/* bad */
h1 {
  font-weight: bold;
}
```

4.5 行高

[建议] line-height 在定义文本段落时，应使用数值。

解释：

将 line-height 设置为数值，浏览器会基于当前元素设置的 font-size 进行再次计算。在不同字号的文本段落组合中，能达到较为舒适的行间间隔效果，避免在每个设置了 font-size 都需要设置 line-height。

当 `line-height` 用于控制垂直居中时，还是应该设置成与容器高度一致。

示例：

```
.container {  
  line-height: 1.5;  
}
```

5 变换与动画

[强制] 使用 `transition` 时应指定 `transition-property`。

示例：

```
/* good */  
.box {  
  transition: color 1s, border-color 1s;  
}  
  
/* bad */  
.box {  
  transition: all 1s;  
}
```

[建议] 尽可能在浏览器能高效实现的属性上添加过渡和动画。

解释：

见[本文](#)，在可能的情况下应选择这样四种变换：

- `transform: translate(npx, npx);`
- `transform: scale(n);`
- `transform: rotate(ndeg);`
- `opacity: 0..1;`

典型的，可以使用 `translate` 来代替 `left` 作为动画属性。

示例：

```

/* good */
.box {
  transition: transform 1s;
}
.box:hover {
  transform: translate(20px); /* move right for 20px */
}

/* bad */
.box {
  left: 0;
  transition: left 1s;
}
.box:hover {
  left: 20px; /* move right for 20px */
}

```

7 响应式

[强制] Media Query 不得单独编排，必须与相关的规则一起定义。

示例：

```

/* Good */
/* header styles */
@media (...) {
  /* header styles */
}

/* main styles */
@media (...) {
  /* main styles */
}

/* footer styles */
@media (...) {
  /* footer styles */
}

```

```
/* Bad */

/* header styles */

/* main styles */

/* footer styles */

@media (...) {
    /* header styles */

    /* main styles */

    /* footer styles */
}
```

[强制] Media Query 如果有多个逗号分隔的条件时，应将每个条件放在单独一行中。

示例：

```
@media
(-webkit-min-device-pixel-ratio: 2), /* Webkit-based browsers */
(min--moz-device-pixel-ratio: 2), /* Older Firefox browsers (prior to Firefox 16) */
(min-resolution: 2dppx), /* The standard way */
(min-resolution: 192dpi) { /* dppx fallback */
    /* Retina-specific stuff here */
}
```

[建议] 尽可能给出在高分辨率设备 (**Retina**) 下效果更佳样式。

8 兼容性

8.1 属性前缀

[强制] 带私有前缀的属性由长到短排列，按冒号位置对齐。

解释：

标准属性放在最后，按冒号对齐方便阅读，也便于在编辑器内进行多行编辑。

示例：

```
.box {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
}
```

8.2 Hack

[建议] 需要添加 **hack** 时应尽可能考虑是否可以采用其他方式解决。

解释：

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。

[建议] 尽量使用 **选择器 hack** 处理兼容性，而非 **属性 hack**。

解释：

尽量使用符合 CSS 语法的 selector hack，可以避免一些第三方库无法识别 hack 语法的问题。

示例：

```
/* IE 7 */  
*:first-child + html #header {  
  margin-top: 3px;  
  padding: 5px;  
}  
  
/* IE 6 */  
* html #header {  
  margin-top: 5px;  
  padding: 4px;  
}
```

[建议] 尽量使用简单的 **属性 hack**。

示例：

```
.box {
  _display: inline; /* fix double margin */
  float: left;
  margin-left: 20px;
}

.container {
  overflow: hidden;
  *zoom: 1; /* triggering hasLayout */
}
```

8.3 Expression

[强制] 禁止使用 **Expression**。

9 命名规范

9.1 CSS命名规则

1. 样式类名全部用小写，首字符必须是字母，禁止数字或其他特殊字符。由以字母开头的小写字母（**a-z**）、数字（**0-9**）、中划线（**-**）组成。
2. 可以是单个单词，也可以是组合单词，要求能够描述清楚模块和元素的含义，使其具有语义化。避免使用 **123456...,red,blue,left,right** 之类的（如颜色、字号大小等）矢量命名，如 **class="left-news"**、**class="2"**，以避免当状态改变时名称失去意义。尽量用单个单词简单描述class名称。
3. 双单词或多单词组合方式：形容词-名词、命名空间-名次、命名空间-形容词-名词。例如：**news-list**、**mod-feeds**、**mod-my-feeds**、**cell-title**

9.2 命名空间

在编码思想上，可以将页面拆分成不同的层级（布局、模块、元件）。

布局: 以语义化的单词layout作为命名空间，例如主栏布局命名 layout-main，只改变layout-命名空间后面的命名，layout始终保留。布局的命名空间为layout-xxx。

模块：页面是由一个或多个模块组成，模块的英文单词是module，规范简写成mod，如新闻模块 **mod-news**，照片展示模块 **mod-photo-show**。模块的命名空间为 **mod-xxx**。

元件：元件是属于模块内部的，也可以说模块是由元件和它内部的自有元素组成。如用户照片信息元件 **cell-user-photo**。元件的命名空间为 **cell-xxx**。

9.3 通用命名

9.3.1 样式表文件命名

```
--|
| -- pages(页面样式)
|   |-- page-name.css
|   |-- ...
|
| -- modules(基础模块样式)
|   |-- mod-xxx.css
|   |-- ...
|
| -- base(基础样式)
|   |-- base-xxx.css
|   |-- ...
|
| -- layout(布局, 版面样式)
|   |-- ly-xxx.css
|   |-- ...
|
| -- themes(主题样式)
|   |-- thm-xxx.css
|   |-- ...
|
| -- patch(临时补丁样式)
|   |-- patch-xxx.css
|   |-- ...
|
| -- other(提供额外的组合配置的样式存放点)
```

9.3.2 页面框架命名，一般具有唯一性

ID名称	命名	ID名称	命名
头部	header	主体	main
脚部	footer	容器	wrapper
侧栏	sideBar	栏目	column
布局	layout		

9.3.3 模块结构命名

Class名称	命名	Class名称	命名
模块(如：新闻模块)	mod (mod-news)	标题栏	title
内容	content/container	次级内容	sub-content

9.3.4 导航结构命名

Class名称	命名	Class名称	命名
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

9.3.5 一般元素命名

Class名称	命名	Class名称	命名
二级	sub	面包屑	breadcrumb
标志	logo	广告	bner(禁用banner或ad)
登陆	login	注册	register/reg
搜索	search	加入	join
状态	status	按钮	btn
滚动	scroll	标签页	tab
文章列表	list	短消息	msg/message
当前的	current	提示小技巧	tips
图标	icon	注释	note
指南	guide	服务	service
热点	hot	新闻	news

Class名称	命名	Class名称	命名
下载	download	投票	vote
合作伙伴	partner	友情链接	friendlink
版权	copyright	演示	demo
下拉框	select	摘要	summary
翻页	pages	主题风格	themes
设置	set	成功	suc
按钮	btn	文本	txt
颜色	color/c	背景	bg
边框	border/bor	居中	center
上	top/t	下	bottom/b
左	left/l	右	right/r
添加	add	删除	del
间隔	sp	段落	p
弹出层	pop	公共	global/gb
操作	op	密码	pwd
透明	tran	信息	info
重点	hit	预览	pvw
单行输入框	input	首页	index
日志	blog	相册	photo
留言板	guestbook	用户	user
确认	confirm	取消	cancel
报错	error	登录条	loginbar
标签	tags	加入	joinus

9.3.6 全局皮肤样式

9.3.6.1 文字颜色(命名空间: text-xxx)

text-warn, text-alert...

9.3.6.2 背景颜色(命名空间: bg-xxx)

bg-warn, bg-alert...

9.3.6.3 边框颜色(命名空间: **border-xxx**)

border-warn, border-alert...

其余的样式按照这种方式组织