CSS 编码标准

目录
结构
Selectors
属性
属性排序
供应商前缀
Values
Media Queries
Commenting
最佳实践
WP 管理后台 CSS
相关链接

与所有编码标准一样， WordPress CSS 编码标准旨在为 WordPress
开源项目及社区的各个层面（从核心代码到主题再到插件）建立协作与审查的基准。项目中的文件应呈现出由
单一实体创建的样貌。最重要的是，要编写出可读、有意义、一致且优美的代码。

在核心样式表中，常会发现不一致之处。我们正在努力解决这些问题，并尽力确保此后的补丁和提交遵循 CSS 编码标准。关于上述内容及参与 UI/前端开发的更多信息，将在另一套指南中提供。

结构
样式表有多种不同的组织方法。对于核心 CSS
而言，保持高度的可读性至关重要。这有助于后续贡献者清晰理解文档的结构脉络。

使用制表符而非空格来缩进每个属性。
在章节之间添加两个空行，在同一章节的代码块之间添加一个空行。
每个选择器应独占一行，以逗号或左花括号结尾。属性 -
值对应独占一行，使用一个制表符缩进并以分号结束。右花括号应左对齐，缩进级别与起始选择器保
持一致。

正确示例：

```
#selector-1,
#selector-2,
#selector-3 {
□background: #fff;
□color: #000;
}
```

□□□□□

```
#selector-1, #selector-2, #selector-3 {
□background: #fff;
□color: #000;
□}

#selector-1 { background: #fff; color: #000; }
```

# Selectors

With specificity, comes great responsibility. Broad selectors allow us to be efficient, yet can have adverse consequences if not tested. Location-specific selectors can save us time, but will quickly lead to a cluttered stylesheet. Exercise your best judgment to create selectors that find the right balance between contributing to the overall style and layout of the DOM.

- Similar to the WordPress PHP Coding Standards for file names, use lowercase and separate words with hyphens when naming selectors. Avoid camelcase and underscores.
- Use human readable selectors that describe what element(s) they style.
- Attribute selectors should use double quotes around values.
- Refrain from using over-qualified selectors, `div.container` can simply be stated as `.container`.

Correct:

```
#comment-form {
□margin: 1em 0;
}

input[type="text"] {
□line-height: 1.1;
}
```

Incorrect:

```
#commentForm { /&042; Avoid camelcase. &042;/
  margin: 0;
}

#comment_form { /&042; Avoid underscores. &042;/
  margin: 0;
}

div#comment_form { /&042; Avoid over-qualification. &042;/
  margin: 0;
}

#c1-xr { /&042; What is a c1-xr?! Use a better name. &042;/
  margin: 0;
}

input[type=text] { /&042; Should be [type="text"] &042;/
  line-height: 110% /&042; Also doubly incorrect &042;/
}
```

# 属性

与选择器类似，过于具体的属性会阻碍设计的灵活性。少即是多。确保你没有重复样式或引入固定尺寸（当流体解决方案更可取时）。

- 属性后应跟随冒号和空格。
- 所有属性和值都应小写，字体名称和供应商特定属性除外。
- 颜色使用十六进制代码，或需要透明度时使用 `rgba()`。避免 RGB 格式和大写，并尽可能缩短值：使用 `#fff` 而非 `#FFFFFF`。
- 尽可能使用简写形式，除非覆盖样式，适用于 `background`、`border`、`font`、`list-style`、`margin` 和 `padding` 值。简写参考请见 CSS 简写。

正确示例：

```
#selector-1 {
  background: #fff;
  display: block;
```

```
  margin: 0;
  margin-left: 20px;
}
```

错误示例:

```
#selector-1 {
  background:#FFFFFF;
  display: BLOCK;
  margin-left: 20PX;
  margin: 0;
}
```

# 属性排序

> "将同类属性归组，特别是当属性数量较多时。 " -- Nacin

最重要的是，选择对你有意义且具有语义性的排序方式。随机排序是混乱而非诗意。在 WordPress 核心代码中，我们选择逻辑分组排序，即属性按含义分组，并在组内进行特定排序。组内属性的排序也经过策略性安排，以在各部分之间形成过渡，例如将 `background` 直接置于 `color` 之前。排序的基准顺序是:

- 显示属性
- 定位属性
- 盒模型属性
- 颜色与排版属性
- 其他属性

核心代码中尚未使用的特性（如 CSS3 动画）可能没有预设位置，但通常能以逻辑方式归入上述某一类别。正如 CSS 在不断演进，我们的标准也将随之发展。

相关属性（如 `margin`）应遵循上 /右 /下 /左（ TRBL）顺序，这与属性值的书写顺序一致。角点描述符（如 `border-radius-*-*`）应按左上、右上、右下、左下的顺序排列。这源于简写属性的值顺序规则。

示例:

```
#overlay {
  position: absolute;
  z-index: 1;
  padding: 10px;
```

```
  background: #fff;
  color: #777;
}
```

另一种常用方法（包括 Automattic/WordPress.com 主题团队采用的方法）是按字母顺序排列属性，可选择性地设置例外规则。

示例：

```
#overlay {
  background: #fff;
  color: #777;
  padding: 10px;
  position: absolute;
  z-index: 1;
}
```

## 供应商前缀

更新于 2014-02-13，在 [27174] 之后：

我们使用 Autoprefixer 作为预提交工具来轻松管理必要的浏览器前缀，这使得本节的大部分内容变得无关紧要。对于那些希望在不使用 Grunt 的情况下遵循其输出的人，供应商前缀应按从最长（-webkit-）到最短（无前缀）的顺序排列。所有其他间距规则与其余标准保持一致。

```
.sample-output {
  -webkit-box-shadow: inset 0 0 1px 1px #eee;
  -moz-box-shadow: inset 0 0 1px 1px #eee;
  box-shadow: inset 0 0 1px 1px #eee;
}
```

# Values

There are numerous ways to input values for properties. Follow the guidelines below to help us retain a high degree of consistency.

- Space before the value, after the colon.
- Do not pad parentheses with spaces.

- Always end in a semicolon.
- Use double quotes rather than single quotes, and only when needed, such as when a font name has a space or for the values of the `content` property.
- Font weights should be defined using numeric values (e.g. `400` instead of `normal`, `700` rather than `bold`).
- 0 values should not have units unless necessary, such as with `transition-duration`.
- Line height should also be unit-less, unless necessary to be defined as a specific pixel value. This is more than just a style convention, but is worth mentioning here. More information: https://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/.
- Use a leading zero for decimal values, including in `rgba()`.
- Multiple comma-separated values for one property should be separated by either a space or a newline. For better readability newlines should be used for lengthier multi-part values such as those for shorthand properties like `box-shadow` and `text-shadow`, including before the first value. Values should then be indented one level in from the property.
- Lists of values within a value, like within `rgba()`, should be separated by a space.

Correct:

```
.class { /&042; Correct usage of quotes &042;/
  background-image: url(images/bg.png);
  font-family: "Helvetica Neue", sans-serif;
  font-weight: 700;
}

.class { /&042; Correct usage of zero values &042;/
  font-family: Georgia, serif;
  line-height: 1.4;
  text-shadow:
    0 -1px 0 rgba(0, 0, 0, 0.5),
    0 1px 0 #fff;
}

.class { /&042; Correct usage of short and lengthier multi-part values &042;/
  font-family: Consolas, Monaco, monospace;
  transition-property: opacity, background, color;
  box-shadow:
    0 0 0 1px #5b9dd9,
    0 0 2px 1px rgba(30, 140, 190, 0.8);
}
```

Incorrect:

```
.class { /&042; Avoid missing space and semicolon &042;/
	background:#fff
}

.class { /&042; Avoid adding a unit on a zero value &042;/
	margin: 0px 0px 20px 0px;
}

.class {
	font-family: Times New Roman, serif; /&042; Quote font names when required &042;/
	font-weight: bold; /&042; Avoid named font weights &042;/
	line-height: 1.4em; /&042; Avoid adding a unit for line height &042;/
}

.class { /&042; Incorrect usage of multi-part values &042;/
	text-shadow: 0 1px 0 rgba(0, 0, 0, 0.5),
	                0 1px 0 #fff;
	box-shadow: 0 1px 0 rgba(0, 0,
		0, 0.5),
		0 1px 0 rgba(0,0,0,0.5);
}
```

# Media Queries

Media queries allow us to gracefully degrade the DOM for different screen sizes. If you are adding any, be sure to test above and below the break-point you are targeting.

- It is generally advisable to keep media queries grouped by media at the bottom of the stylesheet.
  - An exception is made for the `wp-admin.css` file in core, as it is very large and each section essentially represents a stylesheet of its own. Media queries are therefore added at the bottom of sections as applicable.
- Rule sets for media queries should be indented one level in.

Example:

```
@media all and (max-width: 699px) and (min-width: 520px) {
	/&042; Your selectors &042;/
}
```

# Commenting

- Comment, and comment liberally. If there are concerns about file size, utilize minified files and the `SCRIPT_DEBUG` constant. Long comments should manually break the line length at 80 characters.
- A table of contents should be utilized for longer stylesheets, especially those that are highly sectioned. Using an index number (`1.0`, `1.1`, `2.0`, etc.) aids in searching and jumping to a location.
- Comments should be formatted much as PHPDoc is. The CSSDoc standard is not necessarily widely accepted or used but some aspects of it may be adopted over time. Section/subsection headers should have newlines before and after. Inline comments should not have empty newlines separating the comment from the item to which it relates.

For sections and subsections:

```
/**
 * #.# Section title
 *
 * Description of section, whether or not it has media queries, etc.
 */

.selector {
	float: left;
}
```

For inline:

```
/* This is a comment about this selector */
.another-selector {
	position: absolute;
	top: 0 !important; /* I should explain why this is so !important */
}
```






如果你试图修复一个问题，尝试先删除代码，而不是添加更多代码。
魔法数字是不吉利的。这些数字是作为一次性快速修复而使用的。例如： .box { margin-top: 37px }。
DOM 会随时间变化，请直接定位你想要使用的元素，而不是通过其父元素来 “找到它 ”
。例如：在元素上使用 .highlight，而不是 .highlight a（其中选择器在父元素上）。
了解何时使用 height
属性。它应该在你包含外部元素（如图像）时使用。否则，为了获得更大的灵活性，请使用 line-
height。
不要重复声明默认的属性和值组合（例如在块级元素上使用 display: block; ）。

WP 管理后台 CSS

查看 WP 管理后台 CSS 审计报告 ，该报告记录了 WP 管理后台 CSS 代码的健康状况。更多详情请参阅 仓库的

README 文件 。

相关链接

编写一致、地道的 CSS 原则： https://github.com/necolas/idiomatic-css。

版本号 #2
由 壳壳虫 创建于 2026-03-27 01:08:34 UTC
由 壳壳虫 更新于 2026-03-27 01:08:34 UTC