# Sass/Less Style Guide

**用更合理的方式写 CSS 和 Sass**

翻译自 [Airbnb CSS / Sass Styleguide](https://github.com/airbnb/css)

## 目录

1. [术语](#terminology) - [规则声明](#rule-declaration) - [选择器](#selectors) - [属性](#properties)
2. [CSS](#css) - [格式](#formatting) - [注释](#comments) - [OOCSS 和 BEM](#oocss-and-bem) - [ID 选择器](#id-selectors) - [JavaScript 钩子](#javascript-hooks) - [边框](#border)
3. [Sass](#sass) - [语法](#syntax) - [排序](#ordering-of-property-declarations) - [变量](#variables) - [Mixins](#mixins) - [扩展指令](#extend-directive) - [嵌套选择器](#nested-selectors)

## 术语

### 规则声明

我们把一个（或一组）选择器和一组属性称之为 “规则声明”。举个例子：

.listing {  
 font-size: 18px;  
 line-height: 1.2;  
}

### 选择器

在规则声明中，“选择器” 负责选取 DOM 树中的元素，这些元素将被定义的属性所修饰。选择器可以匹配 HTML 元素，也可以匹配一个元素的类名、ID, 或者元素拥有的属性。以下是选择器的例子：

.my-element-class {  
 /\* ... \*/  
}  
  
[aria-hidden] {  
 /\* ... \*/  
}

### 属性

最后，属性决定了规则声明里被选择的元素将得到何种样式。属性以键值对形式存在，一个规则声明可以包含一或多个属性定义。以下是属性定义的例子：

/\* some selector \*/ {  
 background: #f1f1f1;  
 color: #333;  
}

## CSS

### 格式

* 使用 2 个空格作为缩进。
* 类名建议使用破折号代替驼峰法。如果你使用 BEM，也可以使用下划线（参见下面的 [OOCSS 和 BEM](#oocss-and-bem)）。
* 不要使用 ID 选择器。
* 在一个规则声明中应用了多个选择器时，每个选择器独占一行。
* 在规则声明的左大括号 { 前加上一个空格。
* 在属性的冒号 : 后面加上一个空格，前面不加空格。
* 规则声明的右大括号 } 独占一行。
* 规则声明之间用空行分隔开。

**Bad**

.avatar{  
 border-radius:50%;  
 border:2px solid white; }  
.no, .nope, .not\_good {  
 // ...  
}  
#lol-no {  
 // ...  
}

**Good**

.avatar {  
 border-radius: 50%;  
 border: 2px solid white;  
}  
  
.one,  
.selector,  
.per-line {  
 // ...  
}

### 注释

* 建议使用行注释 (在 Sass 中是 //) 代替块注释。
* 建议注释独占一行。避免行末注释。
* 给没有自注释的代码写上详细说明，比如：
  + 为什么用到了 z-index
  + 兼容性处理或者针对特定浏览器的 hack

### OOCSS 和 BEM

出于以下原因，我们鼓励使用 OOCSS 和 BEM 的某种组合：

* 可以帮助我们理清 CSS 和 HTML 之间清晰且严谨的关系。
* 可以帮助我们创建出可重用、易装配的组件。
* 可以减少嵌套，降低特定性。
* 可以帮助我们创建出可扩展的样式表。

**OOCSS**，也就是 “Object Oriented CSS（面向对象的CSS）”，是一种写 CSS 的方法，其思想就是鼓励你把样式表看作“对象”的集合：创建可重用性、可重复性的代码段让你可以在整个网站中多次使用。

参考资料：

* Nicole Sullivan 的 [OOCSS wiki](https://github.com/stubbornella/oocss/wiki)
* Smashing Magazine 的 [Introduction to OOCSS](http://www.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/)

**BEM**，也就是 “Block-Element-Modifier”，是一种用于 HTML 和 CSS 类名的\_命名约定\_。BEM 最初是由 Yandex 提出的，要知道他们拥有巨大的代码库和可伸缩性，BEM 就是为此而生的，并且可以作为一套遵循 OOCSS 的参考指导规范。

* CSS Trick 的 [BEM 101](https://css-tricks.com/bem-101/)
* Harry Roberts 的 [introduction to BEM](http://csswizardry.com/2013/01/mindbemding-getting-your-head-round-bem-syntax/)

**示例**

<article class="listing-card listing-card--featured">  
  
 <h1 class="listing-card\_\_title">Adorable 2BR in the sunny Mission</h1>  
  
 <div class="listing-card\_\_content">  
 <p>Vestibulum id ligula porta felis euismod semper.</p>  
 </div>  
  
</article>

.listing-card { }  
.listing-card--featured { }  
.listing-card\_\_title { }  
.listing-card\_\_content { }

* .listing-card 是一个块（block），表示高层次的组件。
* .listing-card\_\_title 是一个元素（element），它属于 .listing-card 的一部分，因此块是由元素组成的。
* .listing-card--featured 是一个修饰符（modifier），表示这个块与 .listing-card 有着不同的状态或者变化。

### ID 选择器

在 CSS 中，虽然可以通过 ID 选择元素，但大家通常都会把这种方式列为反面教材。ID 选择器给你的规则声明带来了不必要的高[优先级](https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity)，而且 ID 选择器是不可重用的。

想要了解关于这个主题的更多内容，参见 [CSS Wizardry 的文章](http://csswizardry.com/2014/07/hacks-for-dealing-with-specificity/)，文章中有关于如何处理优先级的内容。

### JavaScript 钩子

避免在 CSS 和 JavaScript 中绑定相同的类。否则开发者在重构时通常会出现以下情况：轻则浪费时间在对照查找每个要改变的类，重则因为害怕破坏功能而不敢作出更改。

我们推荐在创建用于特定 JavaScript 的类名时，添加 .js- 前缀：

<button class="btn btn-primary js-request-to-book">Request to Book</button>

### 边框

在定义无边框样式时，使用 0 代替 none。

**Bad**

.foo {  
 border: none;  
}

**Good**

.foo {  
 border: 0;  
}

## Sass

### 语法

* 使用 .scss 的语法，不使用 .sass 原本的语法。
* CSS 和 @include 声明按照以下逻辑排序（参见下文）

### 属性声明的排序

1. 属性声明

* 首先列出除去 @include 和嵌套选择器之外的所有属性声明。
* .btn-green {  
   background: green;  
   font-weight: bold;  
   // ...  
   }

1. @include 声明

* 紧随后面的是 @include，这样可以使得整个选择器的可读性更高。
* .btn-green {  
   background: green;  
   font-weight: bold;  
   @include transition(background 0.5s ease);  
   // ...  
   }

1. 嵌套选择器

* \_如果有必要\_用到嵌套选择器，把它们放到最后，在规则声明和嵌套选择器之间要加上空白，相邻嵌套选择器之间也要加上空白。嵌套选择器中的内容也要遵循上述指引。
* .btn {  
   background: green;  
   font-weight: bold;  
   @include transition(background 0.5s ease);  
    
   .icon {  
   margin-right: 10px;  
   }  
   }

### 变量

变量名应使用破折号（例如 $my-variable）代替 camelCased 和 snake\_cased 风格。对于仅用在当前文件的变量，可以在变量名之前添加下划线前缀（例如 $\_my-variable）。

### Mixins

为了让代码遵循 DRY 原则（Don’t Repeat Yourself）、增强清晰性或抽象化复杂性，应该使用 mixin，这与那些命名良好的函数的作用是异曲同工的。虽然 mixin 可以不接收参数，但要注意，假如你不压缩负载（比如通过 gzip），这样会导致最终的样式包含不必要的代码重复。

### 扩展指令

应避免使用 @extend 指令，因为它并不直观，而且具有潜在风险，特别是用在嵌套选择器的时候。即便是在顶层占位符选择器使用扩展，如果选择器的顺序最终会改变，也可能会导致问题。（比如，如果它们存在于其他文件，而加载顺序发生了变化）。其实，使用 @extend 所获得的大部分优化效果，gzip 压缩已经帮助你做到了，因此你只需要通过 mixin 让样式表更符合 DRY 原则就足够了。

### 嵌套选择器

**请不要让嵌套选择器的深度超过 3 层！**

.page-container {  
 .content {  
 .profile {  
 // STOP!  
 }  
 }  
}

当遇到以上情况的时候，你也许是这样写 CSS 的：

* 与 HTML 强耦合的（也是脆弱的）*—或者—*
* 过于具体（强大）*—或者—*
* 没有重用

再说一遍: **永远不要嵌套 ID 选择器！**

如果你始终坚持要使用 ID 选择器（劝你三思），那也不应该嵌套它们。如果你正打算这么做，你需要先重新检查你的标签，或者指明原因。如果你想要写出风格良好的 HTML 和 CSS，你是**不**应该这样做的。