CSS 规范

语法

- 用两个空格来代替制表符（tab） -- 这是唯一能保证在所有环境下获得一致展现的方法。
- 为选择器分组时，将单独的选择器单独放在一行。
- 为了代码的易读性，在每个声明块的左花括号前添加一个空格。
- 声明块的右花括号应当单独成行。
- 每条声明语句的 `:` 后应该插入一个空格。
- 为了获得更准确的错误报告，每条声明都应该独占一行。
- 所有声明语句都应当以分号结尾。最后一条声明语句后面的分号是可选的，但是，如果省略这个分号，你的代码可能更易出错。
- 对于以逗号分隔的属性值，每个逗号后面都应该插入一个空格（例如，`box-shadow`）。
- 不要在 `rgb()`、`rgba()`、`hsl()`、`hsla()` 或 `rect()` 值的*内部*的逗号后面插入空格。这样利于从多个属性值（既加逗号也加空格）中区分多个颜色值（只加逗号，不加空格）。
- 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`）。
- 十六进制值应该全部小写，例如，`#fff`。在扫描文档时，小写字符易于分辨，因为他们的形式更易于区分。
- 尽量使用简写形式的十六进制值，例如，用 `#fff` 代替 `#ffffff`。
- 为选择器中的属性添加双引号，例如，`input[type="text"]`。只有在某些情况下是可选的，但是，为了代码的一致性，建议都加上双引号。
- 避免为 0 值指定单位，例如，用 `margin: 0;` 代替 `margin: 0px;`。

```
/* Bad CSS */
.selector, .selector-secondary, .selector[type=text] {
  padding:15px;
  margin:0px 0px 15px;
  background-color:rgba(0, 0, 0, 0.5);
  box-shadow:0px 1px 2px #CCC,inset 0 1px 0 #FFFFFF
}

/* Good CSS */
.selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
```

```
  margin-bottom: 15px;
  background-color: rgba(0,0,0,.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

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：

1. Positioning
2. Box model
3. Typographic
4. Visual

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。

完整的属性列表及其排列顺序请参考 Recess。

```
.declaration-order {
  /* Positioning */
  position: absolute;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  z-index: 100;

  /* Box-model */
  display: block;
  float: right;
  width: 100px;
  height: 100px;

  /* Typography */
  font: normal 13px "Helvetica Neue", sans-serif;
  line-height: 1.5;
  color: #333;
  text-align: center;

  /* Visual */
  background-color: #f5f5f5;
  border: 1px solid #e5e5e5;
  border-radius: 3px;

  /* Misc */
  opacity: 1;
}
```

简写形式的属性声明

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
- font
- 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。

```
/* Bad example */
.element {
  margin: 0 0 10px;
  background: red;
  background: url("image.jpg");
  border-radius: 3px 3px 0 0;
}

/* Good example */
.element {
  margin-bottom: 10px;
  background-color: red;
  background-image: url("image.jpg");
  border-top-left-radius: 3px;
  border-top-right-radius: 3px;
}
```

Less 和 Sass 中的嵌套

避免不必要的嵌套。这是因为虽然你可以使用嵌套，但是并不意味着应该使用嵌套。只有在必须将样式限制在父元素内（也就是后代选择器），并且存在多个需要嵌套的元素时才使用嵌套。

```
// Without nesting
.table > thead > tr > th { … }
.table > thead > tr > td { … }

// With nesting
.table > thead > tr {
  > th { … }
  > td { … }
}
```

Less 和 Sass 中的操作符

为了提高可读性，在圆括号中的数学计算表达式的数值、变量和操作符之间均添加一个空格。

```
// Bad example
.element {
  margin: 10px 0 @variable*2 10px;
}

// Good example
.element {
  margin: 10px 0 (@variable * 2) 10px;
}
```

注释

代码是由人编写并维护的。请确保你的代码能够自描述、注释良好并且易于他人理解。好的代码注释能够传达上下文关系和代码目的。不要简单地重申组件或 class 名称。

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。

class 命名

- class 名称中只能出现小写字符和破折号（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`）。
- 避免过度任意的简写。`.btn` 代表 *button*，但是 `.s` 不能表达任何意思。
- class 名称应当尽可能短，并且意义明确。
- 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。
- 基于最近的父 class 或基本（base） class 作为新 class 的前缀。
- 使用 `.js-*` class 来标识行为（与样式相对），并且不要将这些 class 包含到 CSS 文件中。

在为 Sass 和 Less 变量命名时也可以参考上面列出的各项规范。

```
/* Bad example */
.t { ... }
.red { ... }
.header { ... }

/* Good example */
.tweet { ... }
.important { ... }
.tweet-header { ... }
```

选择器

- 对于通用元素使用 class ， 这样利于渲染性能的优化。

- 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）。

```
/* Bad example */
span { ... }
.page-container #stream .stream-item .tweet .tweet-header .username { ... }
.avatar { ... }

/* Good example */
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```