编写规范 - HTML & CSS

来源

大致参考http://codeguide.bootcss.com/，有小部分改动。

黄金定律

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。

HTML

语法

- 用 4 个空格代替制表符（tab）作为一次缩进。
- 嵌套元素应当缩进一次。
- 对于属性的定义，确保全部使用双引号，绝不要使用单引号。
- 不在自闭合（self-closing）元素的尾部添加斜线。
- 不省略可选的结束标签（closing tag）（如`</li>`或`</body>`）。

HTML5 doctype

为每个 HTML 页面的第一行添加标准模式（standard mode）的声明（`<!DOCTYPE html>`）

语言属性

为 html 根元素指定`lang`属性，为文档设置正确的语言。

IE 兼容模式

- IE 支持通过特定的`<meta>`标签来确定绘制当前页面所应该采用的 IE 版本。除非有强烈的特殊需求，否则最好是设置为`edge mode`，从而通知 IE 采用其所支持的最新的模式。
- 参考。

```
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
```

字符编码

在`<head>`中明确声明字符编码，确保浏览器快速并容易的判断页面内容的渲染方式，一般采用utf-8。 `<meta charset="UTF-8">`

引入 CSS 和 JavaScript 文件

在引入 CSS 和 JavaScript 文件时一般不需要指定`type`属性。

实用为王

尽量遵循 HTML 标准和语义，但是不要以牺牲实用性为代价。任何时候都要尽量使用最少的标签并保持最小的复杂度。

属性顺序

HTML 属性应当按照以下给出的顺序依次排列，确保代码的易读性。

- class
- id, name
- data-*
- src, for, type, href, value
- title, alt
- role, aria-*

例：

```
<a class="..." id="..." data-toggle="modal" href="#">Example link</a>

<input class="form-control" type="text">

<img src="..." alt="...">
```

布尔（boolean）型属性

- 元素的布尔型属性如果有值，就是`true`，如果没有值，就是`false`。如果属性存在，其值必须是空字符串或 [...] 属性的规范名称，并且不要在首尾添加空白符。

减少标签的数量

- 编写 HTML 代码时，尽量避免多余的父元素。很多时候，这需要迭代和重构来实现。

例：

```
<!-- Not so great -->
<span class="avatar">
    <img src="...">
</span>

<!-- Better -->
<img class="avatar" src="...">
```

JavaScript 生成的标签

- 尽量避免使用 JavaScript 生成标签。

CSS

语法

- 用 4 个空格代替制表符（tab）作为一次缩进。
- 为选择器分组时，将单独的选择器单独放在一行。
- 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。
- 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。
- 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（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。
- 对于属性值或颜色参数，省略小于 1 的小数前面的 0 。
- 十六进制值全部小写。
- 尽量使用简写形式的十六进制值。
- 为选择器中的属性添加双引号，例，`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：

```
/* 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：

- Positioning
- Box model

- Typographic
- Visual

完整的属性列表及其排列顺序请参考 Recess。

例：

```
.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;
}
```

不要使用@import

替代方法有以下几种：

- 使用多个<link>元素
- 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
- 通过 Rails、Jekyll 或其他系统中提供过 CSS 文件合并功能

例：

```
<!-- Use link elements -->
<link rel="stylesheet" href="core.css">

<!-- Avoid @imports -->
<style>
@import url("more.css");
</style>
```

请参考 Steve Souders 的文章了解更多知识。

媒体查询（Media query）的位置

将媒体查询放在尽可能相关规则的附近，不要将他们打包放在一个单一样式文件中或者放在文档底部。

例：

```
.element { ... }
.element-avatar { ... }
.element-selected { ... }

@media (min-width: 480px) {
.element { ...}
.element-avatar { ... }
.element-selected { ... }
}
```

带前缀的属性

当使用特定厂商的带有前缀的属性时，通过缩进的方式，让每个属性的值在垂直方向对齐，这样便于多行编辑。 例：

```
/* Prefixed properties */
.selector {
-webkit-box-shadow: 0 1px 2px rgba(0,0,0,.15);
        box-shadow: 0 1px 2px rgba(0,0,0,.15);
}
```

单行规则声明

对于只包含一条声明的样式，为了易读性和便于快速编辑，建议将语句放在同一行。对于带有多条声明的样式，还是应当将声明分为多行。

例：

```
/* Single declarations on one line */
.span1 { width: 60px; }
.span2 { width: 140px; }
.span3 { width: 220px; }

/* Multiple declarations, one per line */
.sprite {
    display: inline-block;
    width: 16px;
    height: 15px;
    background-image: url(../img/sprite.png);
}
.icon           { background-position: 0 0; }
.icon-home      { background-position: 0 -20px; }
.icon-account   { background-position: 0 -40px; }
```

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。

请阅读参考文章。

例：

```
/* 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。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。 扩展阅读：

- Nesting in Sass and Less

例：

```
// 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

在Less与Sass中，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。

例：

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

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

注释

确保代码能够自描述、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。

例：

```
/* Bad example */
/* Modal header */
.modal-header {
    ...
}

/* Good example */
/* Wrapping element for .modal-title and .modal-close */
.modal-header {
    ...
}
```

class 命名

`class`的命名需要遵循：

- `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。

例：

```
/* 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）。

扩展阅读：

- Scope CSS classes with prefixes
- Stop the cascade

例：

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

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

代码组织

- 以组件为单位组织代码段。
- 制定一致的注释规范。
- 使用一致的空白符将代码分隔成块。
- 如果使用了多个 CSS 文件，将其按照组件而非页面的形式分拆。

例：

```
/*
* Component section heading
*/

.element { ... }


/*
* Component section heading
*
* Sometimes you need to include optional context for the entire component. Do
that up here if it's important enough.
*/

.element { ... }

/* Contextual sub-component or modifer */
```

```
.element-heading { ... }
```

编辑器配置

- 用两个空格代替制表符（soft-tab 即用空格代表 tab 符）。
- 保存文件时，删除尾部的空白符。
- 设置文件编码为 UTF-8。
- 在文件结尾添加一个空白行。