前端编码规范

> Every line of code should appear to be written by a single person, no matter the number of contributors.

命名

命名应尽量准确、达意，避免使用中文拼音、中文拼音首字母命名。

- **项目名**采用全小写单词，多个单词间以中划线分隔。例如: `your-project-name`
- **目录名**采用全小写单词，多个单词间以中划线分隔。如存在多个文件，则采用复数命名。例如: `stylesheets routes`
- **文件名**采用全小写单词，多个单词间以中划线分隔。例如: `code-guide.html code-guide.css code-guide.js`

例如：Vue.js项目结构

```
vue-project
  |-- build
  | |-- ...
  |-- config
  | |-- config.js
  |-- node_modules
  | |-- ...
  |-- static
  |-- src
  | |-- assets
  | | |-- images
  | | | |-- logo.png
  | | | |-- banner.png
  | | |-- stylesheets
  | |   |-- base.css
  | |   |-- index.css
  | |-- components
  | | |-- ...
  | |-- views
  | | |-- app.vue
  | | |-- home.vue
  | | |-- about.vue
  | |-- app.js
  |-- .babelrc
  |-- .editorconfig
  |-- .eslintrc.js
  |-- .gitignore
  |-- index.html
  |-- README.md
```

HTML

- 标签名采用全小写单词。
- 标签缩进2个空格。
- 属性名/属性值采用全小写单词，多个单词间以中划线分隔。

- 不要忽略可选的关闭标签。例如，`</li>` 和 `</body>`
- 根据 HTML5 规范, 通常在引入 CSS 和 JavaScript 时不需要指明 type，因为 `text/css` 和 `text/javascript` 分别是他们的默认值。
- 尽量遵循 HTML 标准和语义，但是不应该以浪费实用性作为代价。任何时候都要用尽量小的复杂度和尽量少的标签来解决问题。
- HTML 属性应该按照特定的顺序出现以保证易读性 `class id name src alt ...`

```
<!-- bad -->
<!DOCTYPE html>
<html lang="zh-CN">
    <head>
        <title>title</title>
        <link type="text/css" rel="stylesheet" href="code-guide.css">
    </head>
<body>
    <img src="logo.png" alt="logo">
    <h1 class="helloWorld" id='id' customAttr='customAttr'>Hello, world!</h1>
    <script type="text/javascript" src="code-guide.js"></script>
</body>
</html>

<!-- good -->
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <title>title</title>
    <link rel="stylesheet" href="code-guide.css">
  </head>
<body>
  <img src="logo.png" alt="logo">
  <h1 class="hello-world" id="id" custom-attr="custom-attr">Hello, world!</h1>
  <script src="code-guide.js"></script>
</body>
</html>
```

CSS

语法

- 使用2个空格缩进
- 使用组合选择器时，保持每个独立的选择器占用一行。
- 为了代码的易读性，在每个声明的左括号前增加一个空格。
- 声明块的右括号 `}` 应该另起一行。
- 每条声明 `:` 后应该插入一个空格。
- 每条声明应该只占用一行来保证错误报告更加准确。
- 所有声明应该以分号结尾。虽然最后一条声明后的分号是可选的，但是如果没有他，你的代码会更容易出错。
- 逗号分隔的取值，都应该在逗号之后增加一个空格。例如: `box-shadow`
- 不要在颜色值 `rgb() rgba() hsl()` 中增加空格，并且不要带有取值前面不必要的 `0` (比如: 使用 `.5` 替代 `0.5`)。
- 所有的十六进制值都应该使用小写字母，例如 `#fff`。因为小写字母有更多样的外形，在浏览文档时，他们能够更轻松的被区分开来。

- 尽可能使用短的十六进制数值，例如使用 `#fff` 替代 `#ffffff`。
- 为选择器中的属性取值添加引号，例如 `input[type="text"]`。 他们只在某些情况下可有可无，所以都使用引号可以增加一致性。
- 不要为 `0` 指明单位，比如使用 `margin: 0;` 而不是 `margin: 0px;`。

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

/* good */
.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

相关的属性声明应该以下面的顺序分组处理:

1. Position 定位
2. Box model 盒模型
3. Typographic 排版
4. Visual 外观

Positioning 处在第一位，因为他可以使一个元素脱离正常文本流，并且覆盖盒模型相关的样式。盒模型紧跟其后，因为他决定了一个组件的大小和位置。

其他属性只在组件 内部 起作用或者不会对前面两种情况的结果产生影响，所以他们排在后面。

```
.declaration-order {
  /* Position */
  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;
}
```

前缀属性

当使用厂商前缀属性时，通过缩进使取值垂直对齐以便多行编辑。

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

Class 命名

- 保持 Class 命名为全小写，可以使用短划线（不要使用下划线和 camelCase 命名）。短划线应该作为相关类的自然间断。(例如: `.btn` 和 `.btn-danger`)。
- 避免过度使用简写。`.btn` 可以很好地描述 button，但是 `.s` 不能代表任何元素。
- Class 的命名应该尽量短，也要尽量明确。
- 命名时使用最近的父节点或者父 class 作为前缀。

```
/* bad */
.t { ... }
.red { ... }
.header { ... }

/* good */
.tweet { ... }
.important { ... }
.tweet-header { ... }
```

选择器

- 使用 classes 而不是通用元素标签来优化渲染性能。
- 避免在经常出现的组件中使用一些属性选择器 (例如: `[class^="..."]`)。浏览器性能会受到这些情况的影响。
- 减少选择器的长度，每个组合选择器选择器的条目应该尽量控制在 3 个以内。
- 只在必要的情况下使用后代选择器 (例如没有使用带前缀 classes 的情况).

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

/* good */
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```

JavaScript

在项目开发中，使用 JavaScript Standard Style 进行代码错误分析。

详细规则见 **Standard rules**

注释

- 代码是由人来编写和维护的。
- 保证你的代码是描述性的，包含好的注释，并且容易被他人理解。
- 好的代码注释传达上下文和目标。

单行注释

- 双斜线后，必须跟注释内容保留1个空格
- 可独占一行, 前边不许有空行, 缩进与下一行代码保持一致

```
// Good
if (condition) {
  ...
}

const name = 'tom'  // 双斜线距离分号2个空格，双斜线后始终保留1个空格
```

多行注释格式

- 最少三行
- 前边留空一行

何时使用

- 难于理解的代码段
- 可能存在错误的代码段
- 浏览器特殊的HACK代码
- 业务逻辑强相关的代码
- 想吐槽的产品逻辑, 合作同事

```
/*
 * 注释内容与星标前保留一个空格
 */
```

文档注释

各类标签 `@param @method` 等，参考 usejsdoc.org

```
/**
 * xxxxxxx
 * @method xxxxxxx
 * @param {Object} xxxxxxx
 * @return {Object} xxxxxxx
 */
```

编辑器

使用 **WebStorm** 作为主要IDE，根据以下设置来配置你的 WebStorm:

- 使用2个空格缩进。
- 在保存时删除尾部的空白字符。
- 设置文件编码为 `UTF-8`。
- 在文件结尾添加一个空白行。

PS: 常用的IDE (WebStorm Sublime VSCode Atom)都支持 editorconfig 插件，如已安装了 editorconfig 插件并在项目中添加了 `.editorconfig` 文件，将这些设置应用到项目的 `.editorconfig` 文件，添加`.editorconfig` 文件后，可不用修改IDE设置。

```
# editorconfig.org

root = true

[*]
charset = utf-8
end_of_line = lf
indent_size = 2
indent_style = space
insert_final_newline = true
trim_trailing_whitespace = true

[*.md]
trim_trailing_whitespace = false
```

了解更多关于 EditorConfig

附录

- 维基百科：HTML
- 维基百科：CSS
- 维基百科：JavaScript
- 为什么文件名要小写？
- 知乎：JavaScript 语句后应该加分号么？
- 知乎：JavaScript 中字符串变量使用单引号和双引号的利弊？
- 知乎：将html5缩写成h5究竟合不合理？
- 不要在CSS中使用`@import`
- IE compatibility mode
- 围观大神撕B