环球易购WEB标准 2.0

前端技术应用规范

2015-5-10

[1. 黄金定律 5](#_Toc31534)

[2. WEB标准是什么？ 5](#_Toc31352)

[3. 采用WEB标准的好处 5](#_Toc19453)

[3.1 节约运营成本 5](#_Toc20982)

[3.2 用户友好性 5](#_Toc22112)

[3.3 加快页面解析速度 5](#_Toc13392)

[2.4 更加方便后期维护 5](#_Toc3686)

[4. 常用技术术语 5](#_Toc17448)

[5. 通用规范 6](#_Toc14279)

[5.1 tab键用四个空格代替 6](#_Toc15208)

[5.2 每个样式属性或者每句代码后加； 6](#_Toc32119)

[6. HTML规范 6](#_Toc22308)

[6.1 语法约定 6](#_Toc5760)

[6.2 HMTL5 doctype 7](#_Toc12150)

[6.3 语言属性 7](#_Toc27360)

[6.4 IE兼容模式 7](#_Toc17263)

[6.5 字符编码 8](#_Toc75)

[6.6 360极速模式 8](#_Toc333)

[6.7 引入CSS和JavaScript文件 8](#_Toc30201)

[6.8 实用为王 8](#_Toc5519)

[6.9 属性顺序 8](#_Toc20460)

[6.10 布尔（boolean）型属性 9](#_Toc24493)

[6.11 JavaScript 生成的标签 9](#_Toc5452)

[6.12 减少标签的数量 9](#_Toc8568)

[6.13 语义化的HTML 9](#_Toc8913)

[7. CSS规范 9](#_Toc22867)

[7.1 CSS常用命名 9](#_Toc19015)

[7.2 CSS文件结构 10](#_Toc20980)

[7.3 语法 11](#_Toc17282)

[7.4 声明顺序 11](#_Toc19587)

[7.5 不要在CSS中直接使用@import 12](#_Toc29234)

[7.6 媒体查询（Media query）的位置 13](#_Toc29187)

[7.7带前缀的属性 13](#_Toc18209)

[7.8 简写形式的属性声明 13](#_Toc5257)

[7.9 Less 和 Sass 中的嵌套 14](#_Toc9919)

[7.10 注释 14](#_Toc7871)

[7.11 class命名规范 14](#_Toc4358)

[7.12 选择器 15](#_Toc14547)

[7.13 代码组织 15](#_Toc23861)

[8. JavaScript 16](#_Toc5538)

[8.1文件格式 16](#_Toc20689)

[8.2库引入 16](#_Toc12882)

[8.3命名规范 16](#_Toc21328)

[8.4缩进 17](#_Toc25540)

[8.5空格 17](#_Toc19532)

[8.6换行 19](#_Toc10366)

[8.7语句 20](#_Toc26681)

[8.8循环 20](#_Toc11007)

[8.9字符串 21](#_Toc8185)

[8.10注释 22](#_Toc32667)

[9. 编辑器配置 22](#_Toc437)

1. **黄金定律**

永远遵循同一套编码规范，我们的目标是：**不管有多少人共同参与项目，一定要确保每一行代码都像是同一个人编写的**

1. **WEB标准是什么？**

WEB标准是一系列标准的集合。页面主要由三部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。对应的标准也分三方面：结构化标准语言主要包括HTML和XML，表现标准语言主要包括CSS，行为标准主要包括对象模型（如W3C DOM）、ECMAScript等。

1. **采用WEB标准的好处**

3.1 节约运营成本

采用WEB标准，书写页面，我们可以做到表现和形式及内容的分离，我们采用HTML5语言来表现数据，用CSS来控制形式，写的好的页面，HTML代码中基本上都是用户要看到的数据，而其他修饰性的东西，全部由CSS控制。这样来页面的体积就大大减少了，带宽上的费用就降低了。

举个例子，如果Sammydress网首页每减少1K，按每天1000万pv技术，那么每天将节约10G的带宽流量。并且通过样式控制所以的页面元素的表现，假如修改网站的整体风格，你只需要修改一下样式表文件，就可以轻松搞定了，大可不必对页面进行重新编写代码和内容填充。

3.2 用户友好性

采用WEB标准开发的页面，结构清晰，页面体积小，浏览器兼容性好。普通用户访问的时候，页面打开速度快，而且不管是那种浏览器，都能够正常访问页面，且页面的结构清晰，要找的数据可以很方便的浏览到。

对搜索引擎来说，一个标准的WEB开发的页面，是做过SEO优化的，对搜索引擎更加友好，被搜索引擎收录的机会更多。

3.3 加快页面解析速度

通过实际测试证明，采用内容与设计分离的结构进行页面设计，可以使浏览器对网页的解析速度大幅提高，相对于老式的内容与设计混编的方法，浏览器在解析中可以更好的解析结构、内容和设计

2.4 更加方便后期维护

统一标准的页面，编程习惯的一致性，更加便于后期工作的维护，也能大大缩短新员工的适应和熟悉代码的时间，有利于团队效率的提高

1. **常用技术术语**

浮动：float

宽：width

高：height

块元素：block

背景：background

无序列表：ul

链接：a

表单：form

图片：img

段落：p

文档类型定义：DTD

字体：font-family

字号：font-size

边框：border

字体对齐：text-align

行高：line-height

字色：color

背景不循环：no-repeat

内边距：padding

外间距：margin

显示方式：display

悬停：hover

文字修饰：text-decorantion

上：up

底：buttom

左：left

右：right

自动：auto

粗体：bold

正常：normal

1. **通用规范**

5.1 tab键用四个空格代替

在不同的工具对tab解析不一样，windows下的tab建是占四个空格的位置，而在linux下会变成占八个空格的位置，建议自己设定了tab键所占的位置长度，保证一致性。

* 1. 每个样式属性或者每句代码后加；

方便压压缩工具“断句”,请为每个样式属性或者每句代码后加“；”。

1. **HTML规范**
   1. 语法约定
2. 嵌套元素必须缩进一次（即四个空格）
3. 对于属性的定义，确保全部使用双引号，绝不使要用单引号
4. 自定义属性一定是data-牵头
5. 不要在自闭合（self-closing）元素的尾部添加斜线 -- HTML5 规范中明确说明这是可选的
6. 不要省略可选的结束标签（closing tag）（例如，</li> 或 </body>）
7. 在页面中尽量避免使用style属性,即style="…"
8. 能以背景形式呈现的图片,尽量写入css样式中
9. 重要图片必须加上alt属性; 给重要的元素和截断的元素加上title
10. 给区块代码及重要功能(比如循环)加上注释, 方便后台添加功能

<!DOCTYPE html>

<html>

<head>

<title>Page title</title>

</head>

<body>

<img src="images/company-logo.png" alt="Company" data-alt="sammydress">

<h1 class="hello-world">Hello, world!</h1>

</body>

</html>

* 1. HMTL5 doctype

所有HTML页面第一行添加标准模式的声明，这样能够确保在每个浏览器中拥有一致的表现。文档类型声明及编码: **统一为html5声明类型<!DOCTYPE html>**

<!DOCTYPE html>

<html>

<head>

</head>

</html>

* 1. 语言属性

根据 HTML5 规范：  
 强烈建议为 html 根元素指定 lang 属性，从而为文档设置正确的语言。这将有助于语音合成工具确定其所应该采用的发音，有助于翻译工具确定其翻译时所应遵守的规则等等。

英语站点：

<html lang="EN">

<!-- ... -->

</html>

法语站点：

<html lang="FR">

<!-- ... -->

</html>

中文站点：

<html lang="zh-CN">

<!-- ... -->

</html>

6.4 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。

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

6.5 字符编码

通过明确声明字符编码，能够确保浏览器快速并容易的判断页面内容的渲染方式。这样做的好处是，可以避免在 HTML 中使用字符实体标记（character entity），从而全部与文档编码一致（采用 UTF-8 编码）。

<head>

<meta charset="UTF-8">

</head>

6.6 360极速模式

360浏览器支持通过设置 <meta> 标签来设置360内核模式，360浏览器会在读取到这个标签后，立即切换对应的内核。

<meta name="renderer content="webkit">

6.7 引入CSS和JavaScript文件

根据HTML5 规范，在引入 CSS 和 JavaScript 文件时一般不需要指定 type 属性，因为 text/css 和 text/javascript 分别是它们的默认值。

<!-- External CSS -->

<link rel="stylesheet" href="code-guide.css">

<!-- In-document CSS -->

<style>

/\* ... \*/

</style>

<!-- JavaScript -->

<script src="code-guide.js"></script>

6.8 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。**

6.9 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
* title, alt
* aria-\*, role

class 用于标识高度可复用组件，因此应该排在首位。id 用于标识具体组件，应当谨慎使用（例如，页面内的书签），因此排在第二位。

<a class="..." id="..." data-modal="toggle" href="#">

Example link

</a>

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

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

6.10 布尔（boolean）型属性

布尔型属性可以在声明时不赋值。XHTML 规范要求为其赋值，但是 HTML5 规范不需要。**简单来说，就是不用赋值。**

<input type="text" disabled>

<input type="checkbox" value="1" checked>

<select>

<option value="1" selected>1</option>

</select>

6.11 JavaScript 生成的标签

通过 JavaScript 生成的标签让内容变得不易查找、编辑，并且降低性能。能避免时尽量避免。对应异步请求创建的HTML代码可以是JavaScript模板引擎工具创建，推荐使用laytpl：<http://laytpl.layui.com/>

6.12 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。请看下面的案例

<!-- Not so great -->

<span class="avatar">

<img src="...">

</span>

<!-- Better -->

<img class="avatar" src="...">

6.13 语义化的HTML

尽可能的使用语言化的HTML，例如标签标题根据重要性用h\*(同一页面只能有一个h1), 段落标记用p, 列表用ul, 内联元素中不可嵌套块级元素;不要使用纯样式标签，例如：b、font和u等，改用css设置。语义上需要强调的文本可以包在strong或em标签里，strong和em有‘强调’的语义，其中strong的默认样式是加粗，而em的默认加粗斜体

1. **CSS规范**
   1. CSS常用命名

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 名称 | 定义 | 名称 | 定义 | 名称 | 定义 | 名称 | 定义 |
| 头： | header | 尾： | footer | logo： | logo | 版本： | copyright |
| 内容块： | content | 栏目块： | column | 结构左： | left | 结构中： | center |
| 结构右： | right | 首页导航： | index-nav | 频道二级 | channel-nav | 导航文字： | nav-text |
| 内容导航： | nav | 内容主导航： | main-nav | 子内容导航： | sub-nav | 边导航： | sidebar |
| 左导航： | left-sidebar | 右导航： | right-sidebar | 广告： | ad | 搜索： | search |
| 关键字： | key-word | 标签： | tag | 菜单： | menu | 滚动： | scroll |
| 列表： | list | 下拉： | drop | 按钮： | btn | 登陆： | login |
| 登陆条： | login-bar | 注册： | reg | 提示信息： | msg | 打印： | print |
| 地图： | map | 功能区： | shop | 标题： | title | 更多： | more |
| 视频： | video | 媒体： | media | 新闻： | news | 热点： | hot |
| 评论： | review | 联系： | contact | 链接： | link | 投票： | vote |
| 服务： | service | 指南： | guild | 描述： | description | 信息： | info |
| 状态： | status | 价格： | price | 地址： | address | 产品： | products |
| 跳转： | jump | 条： | bar | 线： | line | 技巧： | tips |

* 1. CSS文件结构

将网站的所有样式，按照职能分为三大分类：base,common和page。任何一个网页的最终表现都是由这三者共同完成。这三者不是并列结构，而是层叠结构；

**Base层**，位于最底层，base曾具有高度可移值性，不同设计风格的网站可以使用同一个base层。这一层的内容很少，所以可以简单的放在一个文件里，例如base.css。

**这个base.css文件主要可分为两大部分，CSS reset和通用原子类（可以通过挂载直接使用的class）。Base层，相对稳定，基本上不需要维护。**

<div class="f14 pr ml5 mb20 fb fl"></div>

提供原子类有：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **class名称** | **说明** | **class名称** | **说明** | **class名称** | **说明** | **class名称** | **说明** |
| f14 | 字体14px | f16 | 字体16px | f18 | 字体18px | f20 | 字体20px |
| fb | 字体加粗 | fn | 字体不加粗 | lh150 | 行高150% | f180 | 行高180% |
| lh200 | 行高200% | unl | 添加下划线 | tl | 文本左对齐 | tc | 文本居中对齐 |
| tr | 文本右对齐 | fl | 左浮动 | fr | 右浮动 | clearfix | 清除浮动 |
| cb | 清除所以浮动 | cl | 清除左浮动 | cr | 清除右浮动 | vm | 垂直方向居中 |
| pr | 相对定位 | pa | 绝对定位 | zoom | zoom：1 | hidden | 元素不可见 |
| none | 元素隐藏 | w10 | 宽度10px | w20 | 宽度20px | w30 | 宽度30px |
| w40 | 宽度40px | w50 | 宽度50px | w60 | 宽度60px | w70 | 宽度70px |
| w80 | 宽度80px | w90 | 宽度90px | w100 | 宽度100px | w200 | 宽度200px |
| w300 | 宽度300px | w400 | 宽度400px | w500 | 宽度500px | w600 | 宽度600px |
| w700 | 宽度700px | w800 | 宽度800px | w | 宽度100% | h50 | 高度50px |
| h80 | 高度80px | h100 | 高度100px | h200 | 高度200px | h | 高度100% |
| m10 | 外边距10px | m15 | 外边距15px | m30 | 外边距30px | mt5 | 上外边距5px |
| mt10 | 上外边距10px | mt15 | 上外边距15px | mt20 | 上外边距20px | mt30 | 上外边距30px |
| mt50 | 上外边距50px | mt100 | 上外边距100px | mb10 | 下外边距10px | mb15 | 下外边距15px |
| mb20 | 下外边距20px | mb30 | 下外边距30px | mb50 | 下外边距50px | mb100 | 下外边距100px |
| ml5 | 左外边距5px | ml10 | 左外边距10px | ml15 | 左外边距15px | ml20 | 左外边距20px |
| ml30 | 左外边距30px | ml50 | 左外边距50px | ml100 | 左外边距100px | mr5 | 右外边距5px |
| mr10 | 右外边距10px | mr15 | 右外边距15px | mr20 | 右外边距20px | mr30 | 右外边距30px |
| mr50 | 右外边距50px | mr100 | 右外边距100px | p10 | 内边距10px | p15 | 内边距15px |
| p30 | 内边距30px | pt5 | 上内边距5px | pt10 | 上内边距10px | pt15 | 上内边距15px |
| pt20 | 上内边距20px | pt30 | 上内边距30px | pt50 | 上内边距50px | pb5 | 底内边距5px |
| pb10 | 底内边距10px | pb15 | 底内边距15px | pb20 | 底内边距20px | pb30 | 底内边距30px |
| pb50 | 底内边距50px | pb100 | 底内边距100px | pl5 | 左内边距5px | pl10 | 左内边距10px |
| pl15 | 左内边距15px | pl20 | 左内边距20px | pl30 | 左内边距30px | pl50 | 左内边距50px |
| pl100 | 左内边距100px | pr5 | 右内边距5px | pr10 | 右内边距10px | pr15 | 右内边距15px |
| pr20 | 右内边距20px | pr30 | 右内边距30px | pr50 | 右内边距50px | pr100 | 右内边距100px |

**common层**，提供组件级的css类，将页面内的元素拆分成一小块一小块功能和样式相对独立的小“模块”，将大量重复的“模块”的样式放在common层里，网站的公用头部和底部的样式也统一放在common层里。common层是网站级别的，不同的网站有不同的common层，同一个网站只有一个common层。common层是放在一个common.css中的。

**page层,**网站中非高度重用的模块，page层位于最高层，提供页面级的样式，每个页面可以有自己的page层，如果网站不会过于庞大，建议所有page层的代码反正一个page.css文件中，根据页面配上注释，分块写，便于维护.

* 1. 语法
* 用四个空格来代替制表符（tab）。
* 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。
* 每条声明语句的 : 后应该插入一个空格。
* **建议单行书写样式，每个css的属性都写在一行上（带前缀的属性除外）,每行一个属性，会造成css文件太长。上线前压缩css文件去除空格和注释**
* 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; }

7.4 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：

* 位置相关（Psoitioning）
* 和模型相关（Box model）
* 板式相关（Typographic）
* 视觉相关（Visual）
* 杂项（Misc）

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。

.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;

}

**说明：为方便对比说明，示例代码会选择性使用一行一个属性值的情况，后面代码也是这样，跟规范代码又冲突，**

**特此说明，后面代码不在说明**

7.5 不要在CSS中直接使用@import

与 <link> 标签相比，@import 指令要慢很多，不光增加了额外的请求次数，还会导致不可预料的问题。替代办法有以下几种：

* 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

<!-- Use link elements -->

<link rel="stylesheet" href="core.css">

<!-- Avoid @imports -->

<style>

@import url("more.css");

</style>

7.6 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。如果你把他们分开了，将来只会被大家遗忘。下面给出一个典型的实例。

.element { ... }

.element-avatar { ... }

.element-selected { ... }

@media (min-width: 480px) {

.element { ...}

.element-avatar { ... }

.element-selected { ... }

}

7.7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。

/\* Prefixed properties \*/

.selector {

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

box-shadow: 0 1px 2px rgba(0,0,0,.15);

}

7.8 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;

}

7.9 Less 和 Sass 中的嵌套

避免非必要的嵌套。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。**通常我们的嵌套原则是尽可能的控制在3层内。**

// Without nesting

.table > thead > tr > th { … }

.table > thead > tr > td { … }

// With nesting

.table > thead {

> th { … }

> td { … }

}

7.10 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。

/\* Bad example \*/

/\* Modal header \*/

.modal-header {...}

/\* Good example \*/

/\* Wrapping element for .modal-title and .modal-close \*/

.modal-header {...}

7.11 class命名规范

* class 名称中只能出现小写字符和中划线（不在推荐使用驼峰和下划线）。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。（充分利用Sass和less的&简化书写）
* 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 变量命名是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 { ... }

7.12 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个 。
* 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 { ... }

/\* less and sass \*/

.tweet {

&-header { ... }

&-avatar { ... }

}

7.13 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，这样利于扫描较大的文档。
* 如果使用了多个 less或者sass 文件，将其按照组件而非页面的形式分拆，再在页面中包含需要的less文件，因为页面会被重组，而组件只会被移动。

/\*

\* 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 { ... }

1. **JavaScript**

8.1文件格式

文件编码统一为无BOM的UTF-8，UTF-8 编码具有更广泛的适应性。BOM 在使用程序或工具处理文件时可能造成不必要的干扰，书写过程过, 每行代码结束必须有分号;

8.2库引入

原则上PC仅引入jQuery库, 若需引入第三方库, 须与团队其他人员讨论决定 ，M版引入zepto库

8.3命名规范

* 不要在命名中使用“\”,不要把“\_”作为变量名的第一个字符，变量，函数，参数使用Camel命名法
* 标识符应当直观且可以拼读，最好采用英文单词或其组合，便于记忆和阅读。切忌使用汉语拼音来命名。程序中的英文单词一般不会太复杂，用词应当准确
* 标识符的长度在不影响标识符意义的前提下应当尽量精简。当标识符需要由多个单词组成时，应当将单个单词长度向三靠近
* 程序中不要出现仅靠大小写区分的相似的标识符
* 程序中不要出现标识符完全相同的局部变量和全局变量，尽管两者的 作用域不同而不会发生语法错误，但会使人误解
* 用正确的反义词组命名具有互斥意义的变量或相反动作的函数等。如addEvt，delEvt
* 常量使用全部字母大写，单词间下划线分隔的命名方式
* 类使用Pascal命名法（与骆驼命名法类似。只不过骆驼命名法是首字母小写，而帕斯卡命名法是首字母大写）

function TextNode(options) {

}

* 类的方法 / 属性使用 Camel命名法

function TextNode(value, engine) {

this.value = value;

this.engine = engine;

}

TextNode.prototype.clone = function () {

return this;

};

* 类名使用名词

function Engine(options) {

}

* 函数名使用动宾短语

function getStyle(element) {

}

* boolean类型的变量使用 is 或 has 开头

var isReady = false;

var hasMoreCommands = false;

* Promise对象用动宾短语的进行时表达

var loadingData = ajax.get('url');

loadingData.then(callback);

8.4缩进

* 使用 4 个空格做为一个缩进层级，不允许使用 2 个空格 或 tab 字符(或者设置编辑器Tap缩进是4个空格)
* switch下的case和default 必须增加一个缩进层级

// bad

switch (variable) {

case '1':

// do...

break;

case '2':

// do...

break;

default:

// do...

}

// good

switch (variable) {

case '1':

// do...

break;

case '2':

// do...

break;

default:

// do...

}

8.5空格

* 二元运算符两侧必须有一个空格，一元运算符与操作对象之间不允许有空格

var a = !arr.length;

a++;

a = b + c;

* **用作代码块起始的左花括号 { 前必须有一个空格。**

// bad

if (condition){

}

while (condition){

}

function funcName(){

}

// good

if (condition) {

}

while (condition) {

}

function funcName() {

}

* **if / else / for / while / function / switch / do / try / catch / finally 关键字后，必须有一个空格。**

// bad

if(condition) {

}

while(condition) {

}

(function() {

})();

// good

if (condition) {

}

while (condition) {

}

(function () {

})();

* 在对象创建时，属性中的 : 之后必须有空格，: 之前不允许有空格
* **, 和 ; 前不允许有空格**
* **在函数调用、函数声明、括号表达式、属性访问、if / for / while / switch / catch 等语句中，() 和 [] 内紧贴括号部分不允许有空格**

// bad

callFunc( param1, param2, param3 );

save( this.list[ this.indexes[ i ] ] );

needIncreament && ( variable += increament );

if ( num > list.length ) {

}

while ( len-- ) {

}

// good

callFunc(param1, param2, param3);

save(this.list[this.indexes[i]]);

needIncream && (variable += increament);

if (num > list.length) {

}

while (len--) {

}

8.6换行

* 每个独立语句结束后必须换行
* 在函数声明、函数表达式、函数调用、对象创建、数组创建、for语句等场景中，不允许在 , 或 ; 前换行

// bad

var obj = {

a: 1

, b: 2

, c: 3

};

foo(

aVeryVeryLongArgument

, anotherVeryLongArgument

, callback

);

// good

var obj = {

a: 1,

b: 2,

c: 3

};

foo(

aVeryVeryLongArgument,

anotherVeryLongArgument,

callback

);

* 不同行为或逻辑的语句集，使用空行隔开，更易阅读

function setStyle(element, property, value) {

if (element == null) {

return;

}

element.style[property] = value;

}

8.7语句

* 不得省略语句结束的分号
* 在 if / else / for / do / while 语句中，即使只有一行，也不得省略块 {...}

// bad

if (condition) callFunc();

if (condition)

callFunc();

// good

if (condition) {

callFunc();

}

* 函数定义结束不允许添加分号

// bad

function funcName() {

};

// good

function funcName() {

}

// 如果是函数表达式，分号是不允许省略的。

var funcName = function () {

};

8.8循环

* 要在循环体中包含函数表达式，事先将函数提取到循环体外

// bad

for (var i = 0, len = elements.length; i < len; i++) {

var element = elements[i];

addListener(element, 'click', function () {});

}

// good

function clicker() {

// ......

}

for (var i = 0, len = elements.length; i < len; i++) {

var element = elements[i];

addListener(element, 'click', clicker);

}

* 对循环内多次使用的不变值，在循环外用变量缓存

// good

var width = wrap.offsetWidth + 'px';

for (var i = 0, len = elements.length; i < len; i++) {

var element = elements[i];

element.style.width = width;

// ......

}

// bad

for (var i = 0, len = elements.length; i < len; i++) {

var element = elements[i];

element.style.width = wrap.offsetWidth + 'px';

// ......

}

* 对有序集合进行遍历时，缓存length。

for (var i = 0, len = elements.length; i < len; i++) {

var element = elements[i];

// ......

}

* 对有序集合进行顺序无关的遍历时，使用逆序遍历

var len = elements.length;

while (len--) {

var element = elements[len];

// ......

}

8.9字符串

* 字符串开头和结束使用单引号 '，输入单引号不需要按住shift，方便输入。实际使用中，字符串经常用来拼接HTML。为方便HTML中包含双引号而不需要转义写法

var str = '我是一个字符串';

var html = '<div class="cls">拼接HTML可以省去双引号转义</div>';

$('#id').click(...);

$('.class').bind(...);

* 使用数组或+拼接字符串

// 使用 + 拼接字符串

var str2 = '' // 建议第一个为空字符串, 第二个换行开始并缩进开始, 对齐代码, 方便阅读

str2 += '<ul>';

str2 += '<li>第一项</li>';

str2 += '<li>第二项</li>';

str2 += '</ul>';

* 复杂的数据到视图字符串的转换过程，选用一种模板引擎。推荐使用laytpl（http://laytpl.layui.com/）

8.10注释

* 单行注释，必须独占一行。// 后跟一个空格，缩进与下一行被注释说明的代码一致
* 多行注释，避免使用/\*...\*/这样的多行注释。有多行注释内容时，使用多个单行注释
* 文档化注释,为了便于代码阅读和自文档化，函数必须包含以 /\*\*...\*/ 形式的块注释中（可以使用编辑器插件DocBlockr帮助生成）

/\*\*

\* 函数描述 \*

\* @param {string} p1 参数1的说明

\* @param {string} p2 参数2的说明

\* @param {number=} p3 参数3的说明（可选）

\* @return {Object} 返回值描述

\*/

function foo(p1, p2, p3) {

var p3 = p3 || 10;

return {

p1: p1,

p2: p2,

p3: p3

};

}

1. **编辑器配置**

将你的编辑器按照下面的配置进行设置，以避免常见的代码不一致和差异：

* 用四个空格代替制表符（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。