**项目开发规范文档**

1. 文件规范

1. html, css, js, images文件均归档至约定的目录中;

2. html文件命名: 英文命名, 后缀.html或.htm. 同时将统一页面文件放于同目录中, 以方便后端添加功能时查找对应页面;

3. css文件命名: 英文命名, 后缀.css. 共用base.css, css、重置reset.css 其他页面依实际模块需求命名.;

4. Js文件命名: 英文命名, 后缀.js. 共用main.js, 其他依实际模块需求命名。

文件命名：

**不推荐**

###### html 代码:

1. MyScript.js
2. myCamelCaseName.css
3. i\_love\_underscores.html
4. 1001-scripts.js
5. my-file-min.css

**推荐**

###### html 代码:

1. my-script.js
2. my-camel-case-name.css
3. i-love-underscores.html
4. thousand-and-one-scripts.js
5. my-file.min.css
6. HTML

**Class 与 ID**

* class 应以功能或内容命名，不以表现形式命名；
* class 与 id 单词字母小写，多个单词组成时，采用中划线-分隔；
* 使用唯一的 id 作为 Javascript hook, 同时避免创建无样式信息的 class；

<!-- Not recommended -->

<**div** class="j-hook left contentWrapper"></**div**>

<!-- Recommended -->

<**div** id="j-hook" class="sidebar content-wrapper"></**div**>

**属性顺序**

HTML 属性应该按照特定的顺序出现以保证易读性。

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

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

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

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

**引号**

属性的定义，统一使用双引号。

<!-- Not recommended -->

<**span** id='j-hook' class=text>Google</**span**>

<!-- Recommended -->

<**span** id="j-hook" class="text">Google</**span**>

**严格嵌套约束**

* inline-Level 元素，仅可以包含文本或其它 inline-Level 元素;
* <a>里不可以嵌套交互式元素<a>、<button>、<select>等;
* <p>里不可以嵌套块级元素<div>、<h1>~<h6>、<p>、<ul>/<ol>/<li>、<dl>/<dt>/<dd>、<form>等。

## 布尔值属性

HTML5 规范中 disabled、checked、selected 等属性不用设置值。

<**input** type="text" disabled>

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

<**select**>

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

</**select**>

**语义化**

没有 CSS 的 HTML 是一个语义系统而不是 UI 系统。

通常情况下，每个标签都是有语义的，所谓语义就是你的衣服分为外套， 裤子，裙子，内裤等，各自有对应的功能和含义。所以你总不能把内裤套在脖子上吧。-- 一丝

此外语义化的 HTML 结构，有助于机器（搜索引擎）理解，另一方面多人协作时，能迅速了解开发者意图。

**常见标签语义**

| **标签** | **语义** |
| --- | --- |
| <p> | 段落 |
| <h1> <h2> <h3> ... | 标题 |
| <ul> | 无序列表 |
| <ol> | 有序列表 |
| <blockquote> | 大段引用 |
| <cite> | 一般引用 |
| <b> | 为样式加粗而加粗 |
| <storng> | 为强调内容而加粗 |
| <i> | 为样式倾斜而倾斜 |
| <em> | 为强调内容而倾斜 |
| code | 代码标识 |
| abbr | 缩写 |

**示例**

将你构建的页面当作一本书，将标签的语义对应的其功能和含义；

* 书的名称：<h1>
* 书的每个章节标题: <h2>
* 章节内的文章标题: <h3>
* 小标题/副标题: <h4> <h5> <h6>
* 章节的段落: <p>

### HEAD

**字符编码**

* 以无 BOM 的 utf-8 编码作为文件格式;
* 指定字符编码的 meta 必须是 head 的第一个直接子元素；

<html>

<**head**>

<**meta** charset="utf-8">

......

</**head**>

<**body**>

......

</**body**>

</**html**>

**IE 兼容模式**

优先使用最新版本的IE 和 Chrome 内核

<**meta** http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

**SEO 优化**

<**head**>

<**meta** charset="utf-8">

<**meta** http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<!-- SEO -->

<**title**>Style Guide</**title**>

<**meta** name="keywords" content="your keywords">

<**meta** name="description" content="your description">

<**meta** name="author" content="author,email address">

</**head**>

**iOS 图标**

* apple-touch-icon 图片自动处理成圆角和高光等效果;
* apple-touch-icon-precomposed 禁止系统自动添加效果，直接显示设计原图;

<!-- iPhone 和 iTouch，默认 57x57 像素，必须有 -->

<**link** rel="apple-touch-icon-precomposed" href="/apple-touch-icon-57x57-precomposed.png">

<!-- iPad，72x72 像素，可以没有，但推荐有 -->

<**link** rel="apple-touch-icon-precomposed" href="/apple-touch-icon-72x72-precomposed.png" sizes="72x72">

<!-- Retina iPhone 和 Retina iTouch，114x114 像素，可以没有，但推荐有 -->

<**link** rel="apple-touch-icon-precomposed" href="/apple-touch-icon-114x114-precomposed.png" sizes="114x114">

<!-- Retina iPad，144x144 像素，可以没有，但推荐有 -->

<**link** rel="apple-touch-icon-precomposed" href="/apple-touch-icon-144x144-precomposed.png" sizes="144x144">

## HEAD 模板

<!DOCTYPE html>

<**html** lang=en">

<**head**>

<**meta** charset="utf-8">

<**meta** http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<**title**>Style Guide</**title**>

<**meta** name="description" content="不超过150个字符">

<**meta** name="keywords" content="">

<**meta** name="author" content="name, email@gmail.com">

<!-- 为移动设备添加 viewport -->

<**meta** name="viewport" content="width=device-width, initial-scale=1.0">

<!-- iOS 图标 -->

<**link** rel="apple-touch-icon-precomposed" href="/apple-touch-icon-57x57-precomposed.png">

<**link** rel="alternate" type="application/rss+xml" title="RSS" href="/rss.xml" />

<**link** rel="shortcut icon" href="path/to/favicon.ico">

</**head**>

<body>

//内容区域

</body>

</html>

1. CSS

**代码组织**

* 以组件为单位组织代码段；
* 制定一致的注释规范；
* 组件块和子组件块以及声明块之间使用**一空行**分隔，
* 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，因为页面会被重组，而组件只会被移动；

良好的注释是非常重要的。请留出时间来描述组件（component）的工作方式、局限性和构建它们的方法。不要让你的团队其它成员 来猜测一段不通用或不明显的代码的目的。

提示：通过配置编辑器，可以提供快捷键来输出一致认可的注释模式。

/\* ==========================================================================

组件块

============================================================================ \*/

/\* 子组件块

============================================================================ \*/

.selector {

**padding**: 15px;

**margin-bottom**: 15px;

}

/\* 子组件块

============================================================================ \*/

.selector-secondary {

**display**: block; /\* 注释\*/

}

**Class 和 ID**

* 使用语义化、通用的命名方式；
* 使用连字符 - 作为 ID、Class 名称界定符，不要驼峰命名法和下划线；
* 避免选择器嵌套层级过多，尽量少于 3 级；
* 避免选择器和 Class、ID 叠加使用；

出于[性能考量](http://www.stevesouders.com/blog/2009/06/18/simplifying-css-selectors/)，在没有必要的情况下避免元素选择器叠加 Class、ID 使用。

元素选择器和 ID、Class 混合使用也违反关注分离原则。如果HTML标签修改了，就要再去修改 CSS 代码，不利于后期维护。

/\* Not recommended \*/

.red {}

.box\_green {}

.page .header .login #username **input** {}

**ul**#example {}

/\* Recommended \*/

#nav {}

.box-video {}

#username **input** {}

#example {}

**声明块格式**

* 选择器分组时，保持独立的选择器占用一行；
* 声明块的左括号 { 前添加一个空格；
* 声明块的右括号 } 应单独成行；
* 声明语句中的 : 后应添加一个空格；
* 声明语句应以分号 ; 结尾；
* 一般以逗号分隔的属性值，每个逗号后应添加一个空格；
* rgb()、rgba()、hsl()、hsla() 或 rect() 括号内的值，逗号分隔，但逗号后不添加一个空格；
* 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和尽量简写，例如，#fff 代替 #ffffff；
* 避免为 0 值指定单位，例如，用 margin: 0; 代替 margin: 0px;；

/\* Not recommended \*/

.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

}

/\* Recommended \*/

.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**

相关属性应为一组，推荐的样式编写顺序

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。盒模型决定了组件的尺寸和位置，因此排在第二位。

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;

**box-sizing**: border-box;

**width**: 100px;

**height**: 100px;

**padding**: 10px;

**border**: 1px solid #e5e5e5;

**border-radius**: 3px;

**margin**: 10px;

**float**: right;

**overflow**: hidden;

/\* Typographic \*/

**font**: normal 13px "Helvetica Neue", sans-serif;

**line-height**: 1.5;

**text-align**: center;

/\* Visual \*/

**background-color**: #f5f5f5;

**color**: #fff;

**opacity**: .8;

/\* Other \*/

**cursor**: pointer;

}

**引号使用**

url() 、属性选择符、属性值使用双引号。

**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。

.element { ... }

.element-avatar { ... }

.element-selected { ... }

@media (max-width: 768px) {

.element { ...}

.element-avatar { ... }

.element-selected { ... }

}

**不要使用 @import**

与 <link> 相比，@import 要慢很多，不光增加额外的请求数，还会导致不可预料的问题。

替代办法：

* 使用多个 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；

**LESS**

**代码组织**

代码按一下顺序组织：

1. @import
2. 变量声明
3. 样式声明

@**import** "mixins/size.less";

@**default**-text-color: #333;

.page {

width: 960px;

margin: 0 **auto**;

}

**@import 语句**

@import 语句引用的文需要写在一对引号内，.less 后缀不得省略。引号使用 ' 和 " 均可，但在同一项目内需统一。

/\* Not recommended \*/

@**import** "mixins/size";

@**import** 'mixins/grid.less';

/\* Recommended \*/

@**import** "mixins/size.less";

@**import** "mixins/grid.less";

**混入（Mixin）**

1. 在定义 mixin 时，如果 mixin 名称不是一个需要使用的 className，必须加上括号，否则即使不被调用也会输出到 CSS 中。
2. 如果混入的是本身不输出内容的 mixin，需要在 mixin 后添加括号（即使不传参数），以区分这是否是一个 className。

/\* Not recommended \*/

.big-text {

font-size: 2em;

}

h3 {

.big-text;

.clearfix;

}

/\* Recommended \*/

.big-text() {

font-size: 2em;

}

h3 {

.big-text(); /\* 1 \*/

.clearfix(); /\* 2 \*/

}

**避免嵌套层级过多**

* 将嵌套深度限制在2级。对于超过3级的嵌套，给予重新评估。这可以避免出现过于详实的CSS选择器。
* 避免大量的嵌套规则。当可读性受到影响时，将之打断。推荐避免出现多于20行的嵌套规

1. Javascript

**注释**

**原则**

* As short as possible（如无必要，勿增注释）：尽量提高代码本身的清晰性、可读性。
* As long as necessary（如有必要，尽量详尽）：合理的注释、空行排版等，可以让代码更易阅读、更具美感。

**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。

**函数/方法注释**

1. 函数/方法注释必须包含函数说明，有参数和返回值时必须使用注释标识。；
2. 参数和返回值注释必须包含类型信息和说明；
3. 当函数是内部函数，外部不可访问时，可以使用 @inner 标识；

/\*\*

\* 函数描述

\*

\* **@param** {string} p1 参数1的说明

\* **@param** {string} p2 参数2的说明，比较长

\* 那就换行了.

\* **@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

};

}

**文件注释**

文件注释用于告诉不熟悉这段代码的读者这个文件中包含哪些东西。 应该提供文件的大体内容, 它的作者, 依赖关系和兼容性信息。如下:

/\*\*

\* @fileoverview Description of file, its uses **and** information

\* about its dependencies.

\* @author user@meizu.com (Firstname Lastname)

\* Copyright 2009 Meizu Inc. All Rights Reserved.

\*/

**命名**

**变量**, 使用 Camel 命名法。

**var** loadingModules = {};

**私有属性、变量和方法**以下划线 \_ 开头。

**var** \_privateMethod = {};

**常量**, 使用全部字母大写，单词间下划线分隔的命名方式。

**var** HTML\_ENTITY = {};

1. **函数**, 使用 Camel 命名法。
2. 函数的**参数**, 使用 Camel 命名法。

**function** **stringFormat**(source) {}

**function** **hear**(theBells) {}

1. **类**, 使用 Pascal 命名法
2. 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**;

};

1. **枚举变量** 使用 Pascal 命名法。
2. **枚举的属性**， 使用全部字母大写，单词间下划线分隔的命名方式。

**var** TargetState = {

READING: 1,

READED: 2,

APPLIED: 3,

READY: 4

};

由多个单词组成的 **缩写词**，在命名中，根据当前命名法和出现的位置，所有字母的大小写与首字母的大小写保持一致。

**function** **XMLParser**() {}

**function** **insertHTML**(element, html) {}

**var** httpRequest = **new** HTTPRequest();

**命名语法**

**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);

## True 和 False 布尔表达式

类型检测优先使用 typeof。对象类型检测使用 instanceof。null 或 undefined 的检测使用 == null。

下面的布尔表达式都返回 false:

* null
* undefined
* '' 空字符串
* 0 数字0

但小心下面的, 可都返回 true:

* '0' 字符串0
* [] 空数组
* {} 空对象

**不要在 Array 上使用 for-in 循环**

for-in 循环只用于 object/map/hash 的遍历, 对 Array 用 for-in 循环有时会出错. 因为它并不是从 0 到 length - 1 进行遍历, 而是所有出现在对象及其原型链的键值。

// Not recommended

**function** **printArray**(arr) {

**for** (**var** key **in** arr) {

print(arr[key]);

}

}

printArray([0,1,2,3]); // This works.

**var** a = **new** Array(10);

printArray(a); // This is wrong.

a = document.getElementsByTagName('\*');

printArray(a); // This is wrong.

a = [0,1,2,3];

a.buhu = 'wine';

printArray(a); // This is wrong again.

a = **new** Array;

a[3] = 3;

printArray(a); // This is wrong again.

// Recommended

**function** **printArray**(arr) {

**var** l = arr.length;

**for** (**var** i = 0; i < l; i++) {

print(arr[i]);

}

}

**二元和三元操作符**

操作符始终写在前一行, 以免**分号的隐式插入**产生预想不到的问题。

**var** x = a ? b : c;

**var** y = a ?

longButSimpleOperandB : longButSimpleOperandC;

**var** z = a ?

moreComplicatedB :

moreComplicatedC;

. 操作符也是如此：

var x = foo.bar().

**do**Something().

**do**SomethingElse();

**条件(三元)操作符 (?:)**

三元操作符用于替代 if 条件判断语句。

// Not recommended

**if** (val != 0) {

**return** foo();

} **else** {

**return** bar();

}

// Recommended

**return** val ? foo() : bar();

1. DIV+CSS命名参考表：

      以下为CSS样式命名与[CSS文件](https://zm8.sm-tc.cn/?src=l4uLj8XQ0IiIiNGblomcjIzK0ZyQktCNipKakdCNyM3RjJeLkpM%3D&uid=c50d1d3823a821fee723157311e65f8a&restype=1&from=derive&depth=2&bu=web&v=1&link_type=12)命名参考表，DIV CSS命名集合：

|  |  |
| --- | --- |
| CSS样式命名 | 说明 |
| 网页公共命名 | |
| #wrapper | 页面外围控制整体布局宽度 |
| #container或#content | 容器,用于最外层 |
| #layout | 布局 |
| #head, #header | 页头部分 |
| #foot, #footer | 页脚部分 |
| #nav | 主导航 |
| #subnav | 二级导航 |
| #menu | 菜单 |
| #submenu | 子菜单 |
| #sideBar | 侧栏 |
| #sidebar\_a, #sidebar\_b | 左边栏或右边栏 |
| #[main](javascript:;) | 页面主体 |
| #tag | 标签 |
| #msg #message | 提示信息 |
| #tips | 小技巧 |
| #vote | 投票 |
| #friendlink | 友情连接 |
| #title | 标题 |
| #summary | 摘要 |
| #loginbar | 登录条 |
| #searchInput | 搜索输入框 |
| #hot | 热门热点 |
| #search | 搜索 |
| #search\_output | 搜索输出和搜索结果相似 |
| #searchBar | 搜索条 |
| #search\_results | 搜索结果 |
| #copyright | 版权信息 |
| #branding | 商标 |
| #logo | 网站LOGO标志 |
| #siteinfo | 网站信息 |
| #siteinfoLegal | 法律声明 |
| #siteinfoCredits | 信誉 |
| #joinus | 加入我们 |
| #partner | 合作伙伴 |
| #service | 服务 |
| #regsiter | 注册 |
| arr/arrow | 箭头 |
| #guild | 指南 |
| #sitemap | 网站地图 |
| #list | 列表 |
| #homepage | 首页 |
| #subpage | 二级页面子页面 |
| #tool, #toolbar | 工具条 |
| #drop | 下拉 |
| #dorpmenu | 下拉菜单 |
| #status | 状态 |
| #scroll | 滚动 |
| .tab | 标签页 |
| .left .right .center | 居左、中、右 |
| .news | 新闻 |
| .download | 下载 |
| .banner | 广告条(顶部广告条) |
| 电子贸易相关 | |
| .products | 产品 |
| .products\_prices | 产品价格 |
| .products\_description | 产品描述 |
| .products\_review | 产品评论 |
| .editor\_review | 编辑评论 |
| .news\_release | 最新产品 |
| .publisher | 生产商 |
| .screenshot | 缩略图 |
| .faqs | 常见问题 |
| .keyword | 关键词 |
| .blog | 博客 |
| .forum | 论坛 |
|  |  |