一、 命名规范

1. 项目命名：全部采用小写方式， 以下划线分隔，例：my\_project\_name
2. 目录命名：参照项目命名规则；有复数结构时，要采用复数命名法。例：scripts, styles
3. JS文件命名：参照项目命名规则。例：account\_model.js
4. CSS, SCSS文件命名：参照项目命名规则。例：retina\_sprites.scss
5. HTML文件命名：参照项目命名规则。例：error\_report.html
6. CSS命名：B   
   通用类：使用小写字母，以中划线分隔。例：element-content

业务类：BEM思想，block\_\_element--modifier。例如：person\_\_hand--left

另外：SCSS中的变量、函数、混合、placeholder采用驼峰式命名

1. ID命名：采用驼峰式命名。例：colorBlack

二、 HTML规范

2.1 html头部声明统一

1、 DTD声明

**<!DOCTYPE html>**

2、页面编码统一

<meta charset="UTF-8"/>

3、meta声明

PC端注明: IE兼容/keywords/description

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

<meta name="title" content="优酷-中国领先视频网站,提供视频播放,视频发布,视频搜索 - 优酷视频">

<meta name="keywords" content="视频,视频分享,视频搜索,视频播放,优酷视频">

<meta name="description" content="视频服务平台,提供视频播放,视频发布,视频搜索,视频分享">

4、引入CSS, JS

根据HTML5规范, 通常在引入CSS和JS时不需要指明 type，因为 text/css 和 text/javascript 分别是他们的默认值

HTML5 规范链接：使用link，使用style，使用script

<!-- External CSS -->

<link rel="stylesheet" href="code\_guide.css">

<!-- In-document CSS -->

<style>

...

</style>

<!-- External JS -->

<script src="code\_guide.js"></script>

<!-- In-document JS -->

<script>

...

</script>

2.2 风格与易读性

1、格式缩进

html编码统一格式化显示，使用一个Tab键进行分层缩进 (2个空格宽度)，使整个页面结构层次清晰，方便阅读和修改。

2、模块注释

html较大独立模块之间注释格式统一使用：

<!-- start: XXX模块 -->

<!-- end: XXX模块 -->

或者：

<!-- XXX模块 -->

<!-- /XXX模块 -->

2.3 标签与属性

1、由于html标签和属性不区别大小写，所有建议都采用小写，尤其是自定义标签和属性名，否定js中取不到，如：

<div data-bgColor="red"></div>

$('div').data('bgColor'); // 取不到，已自动被浏览器转成了data-bgcolor

2、所有html属性必须添加双引号（非单引号）。

// 禁止

<div id=mainframe> 或 <div id='mainframe'>

// 推荐

<div id="mainframe">

3、标签属性顺序

属性应该按照特定的顺序出现以保证易读性；

* class
* id
* name
* data-\*
* src, for, type, href, value , max-length, max, min, pattern
* placeholder, title, alt
* role
* required, readonly, disabled

class是为高可复用组件设计的，所以应处在第一位；

id更加具体且应该尽量少使用，所以将它放在第二位

4、boolean属性

boolean属性指不需要声明取值的属性，XHTML需要每个属性声明取值，但是HTML5并不需要；

boolean属性的存在表示取值为true，不存在则表示取值为false。

<input type="text" disabled>

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

<select>

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

</select>

2.4 杂项

1、所有标签必须采用合理嵌套。

// 禁止

<p><b></p></b>

// 推荐

<p><b></b></p>

// 禁止inline级标签嵌套block级标签

<span><div></div></span>

2、img标签中必须添加alt属性。如：<img src="…" alt="logo" />

3、减少标签数量

在编写HTML代码时，需要尽量避免多余的父节点；

很多时候，需要通过迭代和重构来使HTML变得更少。

<!-- Not well -->

<span class="avatar">

<img src="...">

</span>

<!-- Better -->

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

三、 CSS代码规范

3.1 CSS引用规范

1、所有CSS均为外部调用，不得在页面书写任何内部样式或行内样式；

2、html页面引入样式文件：

统一使用link标签，少用@import（原生import有加载性能问题），sass、less、vue.js等文件使用import命令除外（因为最终前端构建工具会将引入文件编译成一个css文件）。

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

<!-- type="text/css"可以省略，清爽 -->

3.2 CSS注释规范

1、css头部文档注释( 统一加上@charset声明 )，如下：

@**charset** "utf-8";

/\*\*

\* @created : 2017/09/15

\* @author : Mr.Wang

\* @version : v1.0

\* @desc : XX模块

\*\*/

关于"version"，如果对代码有修改更新version版本号，并添加相关注释。

2、内部模块之间注释（建议 @模块英文名，好查找）：

/\* @info 商品信息区 -----------------------------------------------------------\*/

.infoArea{}

/\* 单行注释 \*/

.specArea{}

/\* @price 商品价格区 ----------------------------------------------------------\*/

.price{}

3.3 CSS字符规范

1、缩进：使用soft tab（2个空格）；  
2、分号：每个属性声明末尾都要加分号；  
3、空行:

以下几种情况需要空行

/\*\*

\* 文件最后保留一个空行

\* '}'后最好跟一个空行，包括scss中嵌套的规则

\* 属性之间需要适当的空行，具体见属性声明顺序

\*\*/

.element {

...

}

.dialog {

color: red;

&:after {

...

}

}

4、空格

| **以下几种情况不需要空格** | **以下几种情况需要空格** |
| --- | --- |
| 属性名后 | 属性值前 |
| 多个规则的分隔符','前 | 选择器'>', '+', '~'前后 |
| !important '!'后 | '{'前 |
| 属性值中'('后和')'前 | !important '!'前 |
| 行末不要有多余的空格 | 属性值中的','后 |
| 注释'/\*'后和'\*/'前 | |

/\* 示例代码 \*/

.element,

.dialog {

color: red **!important**;

background-color: rgba(0, 0, 0, .5);

}

5、引号

最外层统一使用双引号；

url的内容要用引号；

属性选择器中的属性值需要引号。

.element:after {

content: "";

background-image: url("logo.png");

}

**li**[data-type="single"] {

...

}

### 3.4 样式中属性

1、排序规则：先外部 > 再自身 > 后内部

**A**. 影响文档流的属性（display, position, float, clear, visibility, **table**-layout等）

**B**. 自身盒模型的属性（width, height, margin, padding, border等）

C. 排版相关属性（font, line-height, text-align, text-indent, vertical-align等）

D. 装饰性属性（color, background, opacity, cursor等）

E. 生成内容的属性（content, list-style, quotes等）

2、属性简写

属性简写需要你非常清楚属性值的正确顺序，而且在大多数情况下并不需要设置属性简写中包含的所有值，所以建议尽量分开声明会更加清晰；

margin 和 padding 相反，需要使用简写；

常见的属性简写包括：font background transition animation

/\* not good \*/

.element {

transition: opacity 1s linear 2s;

}

/\* good \*/

.element {

transition-delay: 2s;

transition-timing-function: linear;

transition-duration: 1s;

transition-property: opacity;

}

### 3.5 杂项

1、不允许有空的规则；

2、元素选择器用小写字母；

3、去掉小数点前面的0；

4、去掉数字中不必要的小数点和末尾的0；

5、属性值'0'后面不要加单位；

6、同个属性不同前缀的写法需要在垂直方向保持对齐，具体参照右边的写法；

7、无前缀的标准属性应该写在有前缀的属性后面；

8、不要在同个规则里出现重复的属性，如果重复的属性是连续的则没关系；

9、不要在一个文件里出现两个相同的规则；

10、用 border: 0; 代替 border: none;；

11、选择器不要超过4层（在scss中如果超过4层应该考虑用嵌套的方式来写）；

12、发布的代码中不要有 @import；

13、尽量少用'\*'选择器。

## 三、 JavaScript规范（jslint/eslint）

### 3.1 JS文件引用

1、引入格式：

脚本语言发展至今，也只有js混的最好了，所以type="text/javascript"类型指定可以省去。

<script src="model.js"> </script>

2、引入位置： body标签内最后部（非body外面）, 减少因载入脚本而造成其他页面内容阻塞的问题（js单线程）。

<html>

<body>

<div>页面内容….</div>

<script src="model.js"></script>

</body>

</html>

3、引入方式：html页面中禁止直接编写js代码，统一使用<script>外部引用方式，以便打包压缩和缓存。

### 3.2 空格

以下几种情况需要空格：

* 二元运算符前后
* 三元运算符'?:'前后
* 代码块'{'前
* 下列关键字前：else, while, catch, finally
* 下列关键字后：if, else, for, while, do, switch, case, try, catch, finally, with, return, typeof
* 单行注释'//'后（若单行注释和代码同行，则'//'前也需要），多行注释'\*'后  
  对象的属性值前
* for循环，分号后留有一个空格，前置条件如果有多个，逗号后留一个空格
* 无论是函数声明还是函数表达式，'{'前一定要有空格
* 函数的参数之间

// not good

**var** a = {

b :1

};

// good

**var** a = {

b: 1

};

// not good

++ x;

y ++;

z = x?1:2;

// good

++x;

y++;

z = x ? 1 : 2;

// not good

**var** a = [ 1, 2 ];

// good

**var** a = [1, 2];

// not good

**var** a = ( 1+2 )\*3;

// good

**var** a = (1 + 2) \* 3;

// no space before '(', one space before '{', one space between function parameters

**var** doSomething = **function**(a, b, c) {

// do something

};

// no space before '('

doSomething(item);

// not good

**for**(i=0;i<6;i++){

x++;

}

// good

**for** (i = 0; i < 6; i++) {

x++;

}

### 3.3 JS注释规则

1、 文件头部注释

/\*\*

\* @created : 2017/09/15

\* @author : Mr.Wang

\* @version : v1.0

\* @desc : 当前js模块功能描述

\* @e.g. : 方法用例，如：$('.title').tip();

\*\*/

#### 3.3.2 方法注释及单行注释

1、对于一个较复杂的方法和函数，可用采用多行注释，以便作详情的描述。

/\*\*

\* 此方法是用于解析tpl模块

\* 通过分析html中结构…

\*\*/

2、单行注释

双斜线后，必须跟一个空格；  
缩进与下一行代码保持一致；  
可位于一个代码行的末尾，与代码间隔一个空格。

**var** funName = **function**(arg1, arg2) {

**this**.arg1 = arg1;

// 单行注释说明(上面添加一空行, //与说明之间空一格)

**this**.arg2 = arg2;

};

### 3.4 命名规则

变量名应由 26 个大小写字母（A..Z，a..z），10 个数字（0..9），和“\_”（下划线）组成。

1、通常, 使用”驼峰式”写法，对象、函数和实例时尤其如此。

// 不推荐

**var** is\_my\_object = {};

**var** is-my-object = {};

// 推荐

**var** isMyObject = {};

2、构造函数或类时使用驼峰式大写

// 不推荐

**var** bad = **new** user({

name: 'nope'

});

// 推荐

**var** good = **new** User({

name: 'nope'

});

3、常量大写，并用下划线分隔，形式如：NAMES\_LIKE\_THIS

4、几种特殊命名情况

// 'ID'在变量名中全大写

**var** goodID;

// 'URL'在变量名中全大写

**var** reportURL;

// 'Android'在变量名中大写第一个字母

**var** AndroidVersion;

// 'iOS'在变量名中小写第一个，大写后两个字母

**var** iOSVersion;

5、jquery对象必须以'$'开头命名

// not good

**var** body = $('body');

// good

**var** $body = $('body');

### 3.5 变量声明

一个函数作用域中所有的变量声明尽量提到函数首部，用一个var声明，不允许出现两个连续的var声明

function doSomethingWithItems(items) {

// use one var

var value = 10,

result = value + 10,

i,

len;

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

result += 10;

}

}

### 3.6 null

（1）适用场景：

* 初始化一个将来可能被赋值为对象的变量
* 与已经初始化的变量做比较
* 作为一个参数为对象的函数的调用传参
* 作为一个返回对象的函数的返回值

（2）不适用场景：

* 不要用null来判断函数调用时有无传参
* 不要与未初始化的变量做比较

// not good

**function** **test**(a, b) {

**if** (b === null) {

// not mean b is not supply

...

}

}

**var** a;

**if** (a === null) {

...

}

// good

**var** a = null;

**if** (a === null) {

...

}

### 3.7 undefined

永远不要直接使用undefined进行变量判断；

使用typeof和字符串'undefined'对变量进行判断。

// not good

**if** (person === undefined) {

...

}

// good

**if** (**typeof** person === 'undefined') {

...

}

### 3.8 编码模式

为了规范代码严谨风格，推荐严格模式(Strict Mode)，即总是在模块顶部声明 'use strict';

(**function**(){

**'use strict'**;

**function** **innerFun**(){

**var** j = 0;

//……

}

});

严格模式的一大目标是显性的抛出错误，让你能更方便更快的调试一些隐性的错误。

**1、防止意外的创建了全局变量。**

非严格模式下，为一个未申明的局部变量赋值时会自动创建一个同名的全局变量，这是Js程序中最容易出现的错误之一，在严格模式下这么做会显性的抛出异常。

// 严格模式下会抛出异常

(**function**() {

some = 'foo';

}());

**2、防止函数中的this指针意外指向全局。**

非严格模式下，函数中未被定义或为空( null or undefined)的this会默认指向全局环境(global)。

window.color = 'red';

**function** **getColor**() {

console.log(**this**.color);

}

// 在严格模式中会报错, 非严格模式中则提示red

getColor();

**3、防止重名。**

当编写大量代码时，对象属性和函数参数很容易一不小心被设置成一个重复的名字。严格模式在这种情况下会显性的抛出错误