国内游事业部前端编码规范

版本号0.0.6

**Js：部分**

1. \*方法及方法的调用后面要记得用”;”关闭；

如：

var fn = Function{}; 或者 fn();

1. \*变量的命名要规范；

js文件我们采用 类型小写+语义名称（首字母大写，后面的驼峰）；

如：

iHeight(int), aImgList(array), oPersonCharacter(object), sName(string), bFlag(boolean)

1. \*html页面中js,css选择器的写法：

js选择器，用J-className;

如：J-switchBtn

css 选择器，用中划线连接

如：wrapper-inner, content-primary

1. \*Cookie的使用要谨慎，不宜滥用；容易导致请求头部过重，报400错误；
2. \*<script>标签在页面引入的时候最好加上编码格式（我们通用的编码是utf-8）;

如：

<script charset=”uft-8” ></script>

1. \*尽量避免用全局变量(避免全局变量的办法就是只出现一个全局变量，如：命名空间)；

如：

var oClassSpace = {};

oClassSpace.sub = object || {};

1. \*对dom元素的操作推荐用css样式，不推荐直接在js中写css样式操作元素；

如：

不推荐

$ele.css({height: ‘20px’, opacity: 1,background: ..});

推荐

css文件中写个同一个类，

.switch{

height: ‘20px’;

opacity: 1;

background: ..;

}

$ele.addClass(‘switch’);

1. \*注释，每个功能要加适当的注释，一个功能模块的话用/\*\*/注释，简单的部分用// 注释即可；
2. （弹框）layer, 以后的开发中，对于新功能开发，建议大家用$.layer（2.1版本）;尽量不要自己写，这里面的api够用（暂时pc端这样做）；
3. \*eval函数（魔鬼）不要使用，对于字符串转化为json的时候请用jQuery.json插件代替；
4. \*尽量不要在js文件中拼接html标签，尽量先将标签写在html中，然后用js控制标签的属性。

##### \*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。

如：

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。

如：

// good

if (condition) {

}

while (condition) {

}

function funcName() {

}

// bad

if (condition){

}

while (condition){

}

function funcName(){

}

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

// good

if (condition) {

}

while (condition) {

}

(function () {

})();

// bad

if(condition) {

}

while(condition) {

}

(function() {

})();

##### 15. \*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。

// good

var obj = {

a: 1,

b: 2,

c: 3

};

// bad

var obj = {

a : 1,

b:2,

c :3

};

##### 16. 函数声明、具名函数表达式、函数调用中，函数名和 ( 之间不允许有空格。

// good

function funcName() {

}

var funcName = function funcName() {

};

funcName();

// bad

function funcName () {

}

var funcName = function funcName () {

};

funcName ();

##### 17. \*\* , 和 ; 前不允许有空格。, 后必须有一个空格。

// good

callFunc(a, b);

// bad

callFunc(a , b) ;

18. \*不同行为或逻辑的语句集，使用空行隔开，更易阅读。

// 仅为按逻辑换行的示例，不代表setStyle的最优实现

function setStyle(element, property, value) {

if (element == null) {

return;

}

element.style[property] = value;

}

19. \*建议不在具体某个业务方法里去绑定事件，最好是在页面的自执行匿名函数的前面独立绑定即可。

// 如：

Bad

Var obj = {

Fn: function (){

$(el).on({

‘click’: function(){},

‘touch’: function(){}

})

}

};

God

$(**function**(){

// 单独

$(el).on({

‘click’: function(){},

‘touch’: function(){}

})

});

20. \*ajax的success回调，必须对数据做容错判断(数据的值是否存在和数据类型做判断)，不能直接就用。

**Html: 部分**

1. \*Html标签一定要关闭，除非是一些但标签除外；

如：<input /> <span></span>

2. \*页面上的id选择器永远不能重复，只能唯一一个；

3. \*根据元素的意义来选择要使用什么标签；

如： 用<h2..>定义头部标题，p 元素来定义文字段落，用 a 元素来定义链接锚点，等等。

有根据有目的地使用 HTML 元素，对于可访问性、代码重用、代码效率来说意义重大。

### 4. \*多媒体回溯

对页面上的媒体而言，像图片、视频、canvas 动画等，要确保其有可替代的接入接口。图片文件我们可采用有意义的备选文本（alt），视频和音频文件我们可以为其加上说明文字或字幕。提供可替代内容对可用性来说十分重要。试想，一位盲人用户如何能知晓一张图片是什么，要是没有 @alt 的话。

（图片的 alt 属性是可不填写内容的，纯装饰性的图片就可用这么做：alt=""）。

5. \*HTML的引号，使用双引号(“”) 而不是单引号(”)；

6. \*为每个 HTML 页面的第一行添加标准模式（standard mode）的声明，这样能够确保在每个浏览器中拥有一致的展现。

### 如： HTML 5

<!DOCTYPE html>

### HTML 4.01 Strict

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

### HTML 4.01 Transitional

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

7. 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。

如：

<!-- External CSS -->

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

<!-- In-document CSS -->

<style>

/\* ... \*/

</style>

<!-- JavaScript -->

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

8. \*\*\*视图里不可以直接写相对路径图片，我们的图片都是在CDN上的。一律用线上图片，或者写到CSS里去

**Css 部分**

1. 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。
2. 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）。
3. 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）。
4. 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。
5. 尽量使用简写形式的十六进制值，例如，用 #fff 代替 #ffffff。
6. 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](http://mathiasbynens.be/notes/unquoted-attribute-values#css)，但是，为了代码的一致性，建议都加上双引号。
7. 避免为 0 值指定单位，例如，用 margin: 0; 代替 margin: 0px;。

如：

/\* 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;

}

### 8. 不要使用 @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

如：

<!-- Use link elements -->

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

<!-- Avoid @imports -->

<style>

@import url("more.css");

</style>

### 9. class 命名

* 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 .J-\* 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。