HTML书写规范

HTML是描述网页结构的超文本标记语言，HTML规范能够使HTML代码风格保持一致，使得HTML更容易理解和维护。

 整体结构

【页面头部】

　　1、文件应以“<!DOCTYPE ......>”首行顶格开始，推荐使用“<!DOCTYPE html>”

<!DOCTYPE html>

　　2、必须声明文档的编码charset，且与文件本身编码保持一致，指定字符编码的 meta 必须是 head 的第一个直接子元素。推荐使用UTF-8编码<meta charset="utf-8">

<meta charset="utf-8">

（1）网页显示字符集

<!--html5格式-->

<meta charset="UTF-8">

<!--简体中文-->

<meta http-equiv="content-type" content="text/css;charset=gb2312">

<!--繁体中文-->

<meta http-equiv="content-type" content="text/css;charset=gbk">

<!--英文-->

<meta http-equiv="content-type" content="text/css;charset=utf-8">

（2）可选标签

公司的版权注释

<!--the site is designed by Bob at 2016.9.24-->

网页制作者信息

<meta name="author" content="123456@qq.com">

网站简介

<meta name="description" content="麦子学院创立于xxx年，学院开设了xxx课程,老师拥有x年的教学经历">

设定网页的到期时间，一旦网页过期，必须到服务器上重新调阅

<meta http-equiv="expires" content="Wed,26 Feb 2017 08:24:00 GMT">

禁止浏览器从本地机缓存中调阅内容

<meta http-equiv="pragma" content="no-cache">

用来防止别人在框架中调用你的页面

<meta http-equiv="Window-target" content="\_top">

自动跳转

<meta http-equiv="refresh" content="5;url=http://www.baidu.com">

网页搜索机器人向导，用来告诉机器人哪些页面需要搜索，哪些页 面不需要搜索

<meta name="robots" content="none">

收藏夹图标

<link rel="Shortcut Icon" href="favicon.icon">

3、根据页面内容和需求填写适当的keywords和description

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

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

4、页面title是不可缺少的一项，title 必须作为 head 的直接子元素，并紧随 charset 声明之后

<head>

<meta charset="UTF-8">

<title>页面标题</title>

</head>

【资源引入】

　　1、保证 favicon 可访问

<link rel="shortcut icon" href="path/to/favicon.ico">//用于显示网页导航栏上的logo

　　2、引入 CSS 和 JavaScript 时无须指明 type 属性

　　3、引入 CSS 时必须指明 rel="stylesheet"

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

　　4、使用link将css文件引入，并置于head中；使用script将js文件引入，并置于body底部

　　5、移动环境或只针对现代浏览器设计的 Web 应用，如果引用外部资源的 URL 协议部分与页面相同，建议省略协议前缀。这是因为使用 protocol-relative URL 引入 CSS，在 IE7/8 下，会发两次请求。是否使用 protocol-relative URL 应充分考虑页面针对的环境

<script src="//s1.bdstatic.com/cache/static/jquery-1.10.2.min\_f2fb5194.js"></script>

【结构优化】

　　1、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

　　2、结构顺序和视觉顺序基本保持一致，按照从上至下、从左到右的视觉顺序书写HTML结构。有时为了便于搜索引擎抓取，也会将重要内容在HTML结构顺序上提前

　　3、结构、表现、行为三者分离，避免内联

　　4、每一个块级元素都另起一行，每一行都使用Tab缩进对齐（head和body的子元素不需要缩进）。删除冗余的行尾空格

　　5、对于内容较为简单的表格，建议将tr写成单行

　　6、可以在大的模块之间用空行隔开，使模块更清晰

【语义化】

　　浏览器会根据标签的语义给定一个默认的样式。判断网页标签语义化是否良好的一个简单方法：去掉样式，看网页结构是否组织良好有序，是否仍然有很好的可读性

　　1、尽可能少地使用无语义标签span和div

　　2、在语义不明显，既可以使用p也可以使用div的地方，尽量用p

　　3、在既可以使用div也可以使用section的地方，尽量用section

　　4、不要使用纯样式标签，如b、u等，而改用CSS设置

代码格式

【命名】

　　1、class 必须单词全字母小写，单词间以 - 分隔

　　2、class 必须代表相应模块或部件的内容或功能，不得以样式信息进行命名

<!-- good -->

<div class="sidebar"></div>

<!-- bad -->

<div class="left"></div>

　　3、同一页面，应避免使用相同的 name 与 id。因为IE7-浏览器会混淆元素的 id 和 name 属性， document.getElementById 可能获得不期望的元素。所以在对元素的 id 与 name 属性的命名需要非常小心

【标签】

　　1、标签名必须使用小写字母

　　2、对于无需自闭合的标签，不需要自闭合

　　3、对 HTML5 中规定允许省略的闭合标签，不允许省略闭合标签

　　4、HTML 标签的使用应该遵循标签的语义，且要符合标签嵌套规则

<!-- good -->

<p>Hello StyleGuide!</p>

<!-- bad -->

<P>Hello StyleGuide!</P>

<!-- good -->

<input type="text" name="title">

<!-- bad -->

<input type="text" name="title"

【注释】

　　采用类似标签闭合的写法，与HTML统一格式；注释文案两头空格，与CSS注释统一格式

　　开始注释：<!-- 注释文案 -->（文案两头空格）

　　结束注释：<!-- /注释文案 -->（文案前加“/”符号，类似标签的闭合）

　　允许只有开始注释

<!-- 头部 -->

<div class="g-hd">

<!-- LOGO -->

<h1 class="m-logo"><a href="#">LOGO</a></h1>

<!-- /LOGO -->

<!-- 导航 -->

<ul class="m-nav">

<li><a href="#">NAV1</a></li>

<li><a href="#">NAV2</a></li>

<!-- 更多导航项 -->

</ul>

<!-- /导航 -->

</div>

<!-- /头部 -->

【属性】

　　1、属性和值全部小写

　　2、属性值必须用双引号包围

　　3、布尔类型的属性，建议不添加属性值

　　4、自定义属性建议以 xxx- 为前缀，推荐使用 data-

　　5、可以省略style标签和script标签的type属性

<!-- good -->

<table cellspacing="0">...</table>

<!-- bad -->

<table cellSpacing="0">...</table>

<!-- good -->

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

<!-- bad -->

<script src='esl.js'></script>

<script src=esl.js></script>

【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

特殊元素

【图片】

　　1、禁止 img 的 src 取值为空，否则会导致部分浏览器重新加载一次当前页面

　　2、为图片添加 alt 属性，提高图片加载失败时的用户体验

　　3、避免为 img 添加不必要的 title 属性，多余的 title 影响看图体验，并且增加了页面尺寸

　　4、为图片添加 width 和 height 属性，以避免页面抖动

<img src="#" alt="#" width="#" height="#">

　　5、有下载需求的图片采用 img 标签实现，无下载需求的图片采用 CSS 背景图实现

　　产品 logo、用户头像、用户产生的图片等有潜在下载需求的图片，以 img 形式实现，能方便用户下载

　　无下载需求的图片，比如：icon、背景、代码使用的图片等，尽可能采用 css 背景图实现

【表单】

　　1、有文本标题的控件使用 label 标签将其与其标题相关联。最好将控件置于 label 内，以减少不必要的 id

<label><input type="checkbox" name="confirm" value="on"> 我已确认上述条款</label>

　　2、使用 button 元素时必须指明 type 属性值。因为button 元素的默认 type 为 submit，如果被置于 form 元素中，点击后将导致表单提交

<button type="submit">提交</button>

<button type="button">取消</button>

————————————————

版权声明：本文为CSDN博主「wujingj」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/sinat_19803127/java/article/details/80524165>

## [css书写规范](https://www.cnblogs.com/makai/p/9002898.html)

在书写css样式的时候总是无意中就写乱了，无论是命名或者是样式的书写顺序，这里做一个总结，提醒自己在书写css的时候时刻注意，大家可以参考哈。

## **1. 样式属性顺序**

单个样式规则下的属性在书写时，应按功能进行分组，组之间需要有一个空行。  
同时要以Positioning Model > Box Model > Typographic > Visual 的顺序书写，提高代码的可读性。

1. Positioning Model 布局方式、位置，相关属性包括：position, top, z-index, display, float等
2. Box Model 盒模型，相关属性包括：width, height, padding, margin，border,overflow
3. Typographic 文本排版，相关属性包括：font, line-height, text-align
4. Visual 视觉外观，相关属性包括：color, background, list-style, transform, animation

## **2. CSS选择器命名规则**

分类式命名法(在前端组件化下尤为重要)

* 1. 布局（grid）（.g-）：将页面分割为几个大块，通常有头部、主体、主栏、侧栏、尾部等！
  2. 模块（module）（.m-）：通常是一个语义化的可以重复使用的较大的整体！比如导航、登录、注册等
  3. 元件（unit）（.u-）：通常是一个不可再分的较为小巧的个体，通常被重复用于各种模块中！比如按钮、输 入框、loading等！
  4. 功能（function）（.f-）：为方便一些常用样式的使用，我们将这些使用率较高的样式剥离出来，按需使用，通常这些选择器具有固定样式表现，比如清除浮动等！不可滥用！
  5. 状态（.z-）：为状态类样式加入前缀，统一标识，方便识别，她只能组合使用或作为后代出现（.u-ipt.z-dis{}，.m-list li.z-sel{}）
  6. javascript(.j-)：.j-将被专用于JS获取节点，请勿使用.j-定义样式
* 不要使用 " \_ " 下划线来命名css  
  能良好的区分javascript变量名  
  输入的时候少按一个shift键  
  浏览器兼容性问题（比如使用\_tips的选择器命名，在IE6是无效的）
* id采用驼峰式命名(不要乱用id)
* scss中的变量、函数、混合、placeholder采用驼峰式命名
* 相同语义的不同类命名方法：  
  直接加数字或字母区分即可（如：.m-list、.m-list2、.m-list3等，都是列表模块，但是是完全不一样的模块）。其他举例：.f-fw0、.f-fw1、.s-fc0、.s-fc1、.m-logo2、.m-logo3、u-btn、u-btn2等等。
* 命名方式(BEM)：类-体（例：g-head）、类-体-修饰符（例：u-btn-active）
* 后代选择器：体-修饰符即可（例：.m-page .cut{}）注：后代选择器不要在页面布局中使用，因为污染的可能性较大；

## **3. 最佳写法**

/\* 这是某个模块 \*/

.m-nav{}/\* 模块容器 \*/

.m-nav li,.m-nav a{}/\* 先共性 优化组合 \*/

.m-nav li{}/\* 后个性 语义化标签选择器 \*/

.m-nav a{}/\* 后个性中的共性 按结构顺序 \*/

.m-nav a.a1{}/\* 后个性中的个性 \*/

.m-nav a.a2{}/\* 后个性中的个性 \*/

.m-nav .z-crt a{}/\* 交互状态变化 \*/

.m-nav .z-crt a.a1{}

.m-nav .z-crt a.a2{}

.m-nav .btn{}/\* 典型后代选择器 \*/

.m-nav .btn-1{}/\* 典型后代选择器扩展 \*/

.m-nav .btn-dis{}/\* 典型后代选择器扩展（状态） \*/

.m-nav .btn.z-dis{}/\* 作用同上，请二选一（如果可以不兼容IE6时使用） \*/

.m-nav .m-sch{}/\* 控制内部其他模块位置 \*/

.m-nav .u-sel{}/\* 控制内部其他元件位置 \*/

.m-nav-1{}/\* 模块扩展 \*/

.m-nav-1 li{}

.m-nav-dis{}/\* 模块扩展（状态） \*/

.m-nav.z-dis{}/\* 作用同上，请二选一（如果可以不兼容IE6时使用） \*/

## **4. 统一语义理解和命名**

布局(.g-)

| **语义** | **命名** | **简写** |
| --- | --- | --- |
| 文档 | doc | doc |
| 头部 | head | hd |
| 主体 | body | bd |
| 尾部 | foot | ft |
| 主栏 | main | mn |
| 主栏子容器 | mainc | mnc |
| 侧栏 | side | sd |
| 侧栏子容器 | sidec | sdc |
| 盒容器 | wrap/box | wrap/box |

模块（.m-）、元件（.u-）

| **语义** | **命名** | **简写** |
| --- | --- | --- |
| 导航 | nav | nav |
| 子导航 | subnav | snav |
| 面包屑 | crumb | crm |
| 菜单 | menu | menu |
| 选项卡 | tab | tab |
| 标题区 | head/title | hd/tt |
| 内容区 | body/content | bd/ct |
| 列表 | list | lst |
| 表格 | table | tb |
| 表单 | form | fm |
| 热点 | hot | hot |
| 排行 | top | top |
| 登录 | login | log |
| 标志 | logo | logo |
| 广告 | advertise | ad |
| 搜索 | search | sch |
| 幻灯 | slide | sld |
| 提示 | tips | tips |
| 帮助 | help | help |
| 新闻 | news | news |
| 下载 | download | dld |
| 注册 | regist | reg |
| 投票 | vote | vote |
| 版权 | vcopyright | cprt |
| 结果 | result | rst |
| 标题 | title | tt |
| 按钮 | button | btn |
| 输入 | input | ipt |

功能（.f-）

| **语义** | **命名** | **简写** |
| --- | --- | --- |
| 清除浮动 | clearboth | cb |
| 左浮动 | floatleft | fl |
| 内联 | inlineblock | ib |
| 文本居中 | textaligncenter | tac |
| 垂直居中 | verticalalignmiddle | vam |
| 溢出隐藏 | overflowhidden | oh |
| 完全消失 | displaynone | dn |
| 字体大小 | fontsize | fz |
| 字体粗细 | fontweight | fs |

皮肤（.s-）

| **语义** | **命名** | **简写** |
| --- | --- | --- |
| 字体颜色 | fontcolor | fc |
| 背景颜色 | backgroundcolor | bgc |
| 边框颜色 | bordercolor | bdc |

状态(.z-)

| **语义** | **命名** | **简写** |
| --- | --- | --- |
| 选中 | selected | sel |
| 当前 | current | crt |
| 显示 | show | show |
| 隐藏 | hide | hide |
| 打开 | open | open |
| 关闭 | close vclose |  |
| 出错 | error | err |
| 不可用 | disabled | dis |

## **5. 注意事项**

1. 一律小写，中划线
2. 尽量不用缩写
3. 不要随便使用id
4. 去掉小数点前面的0： 0.9rem => .9rem
5. 使用简写：margin： 0 1rem 3rem

# javascript规范

## 一、规范目的

为提高团队协作效率，便于前端后期优化维护，输出高质量的文档。

## 二、基本准则

* 符合web标准，结构表现行为分离，兼容性优良。页面性能方面，代码要求简洁明了有序， 尽可能的减小服务器负载，保证最快的解析速度。
* 项目的维护和二次开发可能是直接或间接的团队合作，所以创建易维护的代码是一个项目成功与否的关键，易维护的代码意味着具有如下特性：
  + 阅读性好：如良好的注释和命名规范，有文档
  + 具有一致性：看起来如同一个人编写
  + 代码的松耦合，高度模块化：将页面内的元素视为一个个模块，相互独立，尽量避免耦合过高的代码，从html,css,js三个层面都要考虑模块化
  + 严格按照规范编写代码

## 三、命名规范

### 1. 目的

提高代码可预测性和可维护性的方法是使用命名约定，这就意味着采用一致的方法来对变量和函数进行命名。

### 2. 变量名

变量名包括全局变量，局部变量，类变量，函数参数

### 3. 构造函数（类）命名

首字母大写，驼峰式命名。

JS中没有类，但是可以用new调用构造函数：var man = new Person();

### 4. 普通变量命名

首字母小写，驼峰式命名，匈牙利命名

如：nCheckCount 表示整形的数值

### 5. 匈牙利命名法

匈牙利命名法语法：变量名＝类型＋对象描述

* 类型指变量的类型
* 对象描述指对象名字全称或名字的一部分，要求有明确含义，命名要容易记忆容易理解。

****提示****: 虽然JavaScript变量表面上没有类型，但是JavaScript内部还是会为变量赋予相应的类型



### 6. 其他前缀规范

可根据团队及项目需要增加

$：表示Jquery对象

例如：$Content，$Module，一种比较广泛的Jquery对象变量命名规范。

fn：表示函数

例如：fnGetName，fnSetAge；和上面函数的前缀略有不同，改用fn来代替，个人认为fn能够更好的区分普通变量和函数变量。

### 7. 例外情况

以根据项目及团队需要，设计出针对项目需要的前缀规范，从而达到团队开发协作便利的目的。

* 作用域不大临时变量可以简写，比如：str，num，bol，obj，fun，arr。
* 循环变量可以简写，比如：i，j，k等。
* 某些作为不允许修改值的变量认为是常量，全部字母都大写。例如：COPYRIGHT，PI。常量可以存在于函数中，也可以存在于全局。必须采用全大写的命名，且单词以\_分割，常量通常用于ajax请求url，和一些不会改变的数据。

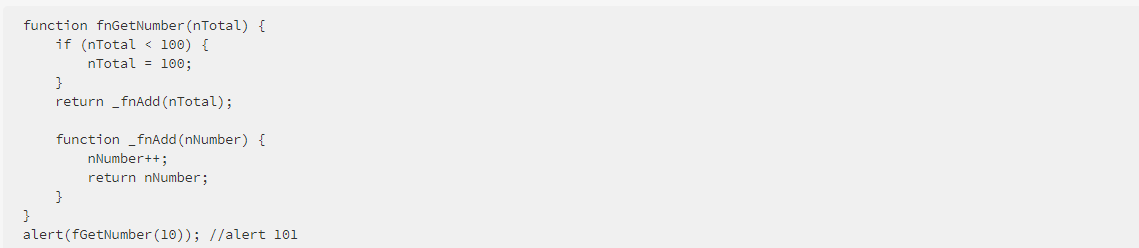
### 8. 函数命名

****普通函数****：首字母小写，驼峰式命名，统一使用动词或者动词+名词形式

例如：fnGetVersion()，fnSubmitForm()，fnInit()；涉及返回逻辑值的函数可以使用is，has，contains等表示逻辑的词语代替动词，例如：fnIsObject()，fnHasClass()，fnContainsElment()。

****内部函数****：使用\_fn+动词+名词形式，内部函数必需在函数最后定义。

例如：



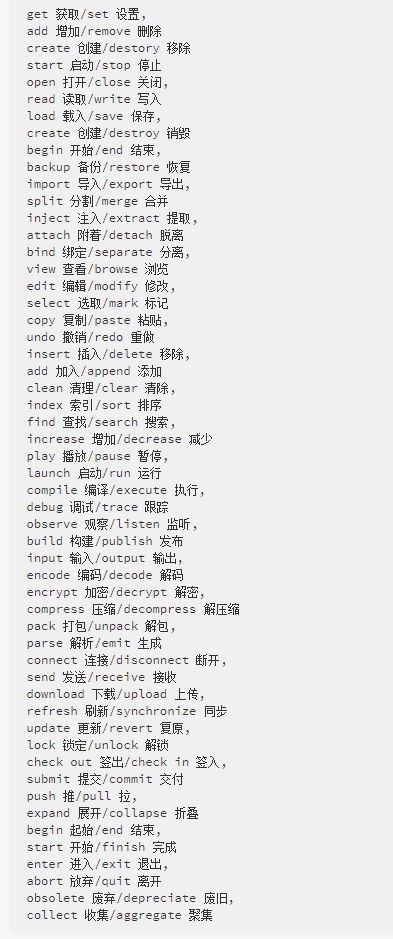
****对象方法与事件响应函数****：对象方法命名使用fn+对象类名+动词+名词形式；

例如： fnAddressGetEmail()，

****事件响应函数****：fn+触发事件对象名+事件名或者模块名

例如：fnDivClick()，fnAddressSubmitButtonClick()

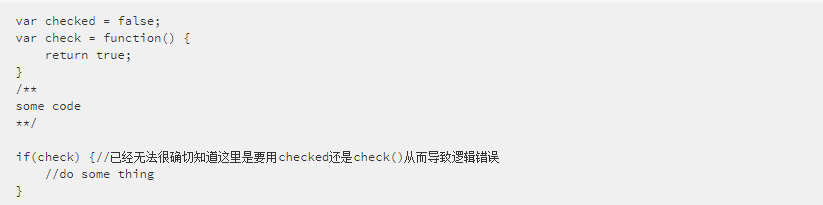
****函数方法常用的动词****：



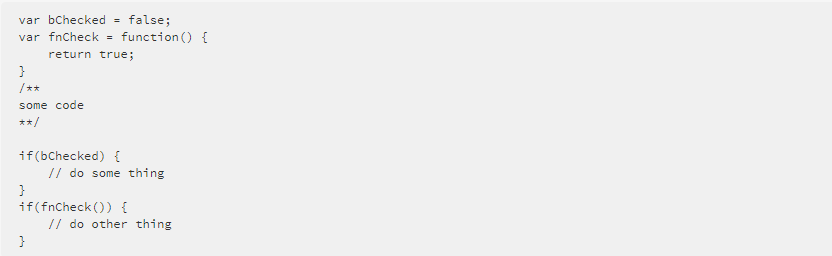
### 9. 变量命名例子

为什么需要这样强制定义变量前缀？正式因为javascript是弱语言造成的。在定义大量变量的时候，我们需要很明确的知道当前变量是什么属性，如果只通过普通单词，是很难区分的。

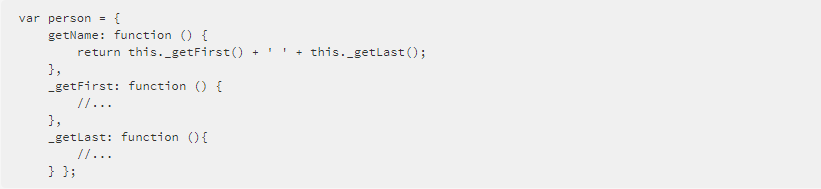
普通代码



规范后代码



如何标明私有方法或私有属性？



在这个例子中，getName()以为这这是API的一个公开的方法，而\_getFirst()和\_getLast()意味着这是一个私有函数。尽管他们都是普通的公开方法，但是使用下划线前缀的表示方法可以提醒使用person对象的用户，告诉他们这些方法在其他地方不能确保一定能够正常工作，不能直接调用。

总结：下面是一些使用下划线约定的变量

* 使用下划线结尾来表明是私有变量，例如name\_和getElements\_()。
* 使用一个下划线前缀来表示受保护属性，使用两个下划线前缀来表示私有属性。

## 四、编写注释

为代码编写注释是非常重要的。通常人们在深入思考一个问题时，会非常清楚这段代码的工作原理。但是当过一周后再次回到该代码时，可能会花上很长时间来回想起那段代码到底是干什么的。

公共组件维护者和各栏目WD都需要在文件头部加上注释说明：



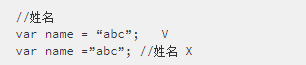
大的模块注释方法：



小的注释；



注释单独一行，不要在代码后的同一行内加注释。例如：



## 五、使用空格

使用空格有助于改善代码的可读性和一致性。在撰写英文文章时在逗号和区间范围后面使用空格。在javascript采用同样的逻辑，可在列表表达式（等价于逗号）和语句结束（等价于完成一次“思考”）后面添加空格。

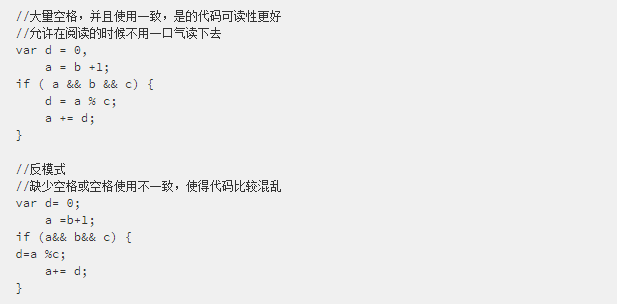
****用处一****

* 在分开for循环的各个部分的分号之后：例如，for (var i = 0; i < 10; i +=1){…}
* 在for循环中初始化多个变量（i和最大值等）：for (var i = 0, max = 10; i < max; i += 1){…}
* 在限定数组项的逗号后面：var a = [1, 2, 3];
* 对象属性的逗号之后和将属性名和属性值分开的冒号之后：var o = {a: 1, b: 2};
* 分隔开函数中各个参数的逗号之后：myFunc(a, b, c)
* 在函数声明的大括号之前：function myFunc() {}
* 在匿名函数表达式之后：var myFunc = function () {};

****用处二****

空格的另外一个很好的用途是用来分隔所有的操作符和操作，这也就是意味着在 +, -, \*, =, <, >, <=, >=, ===, !==, &&, ||, += 等之后使用空格：

例子：

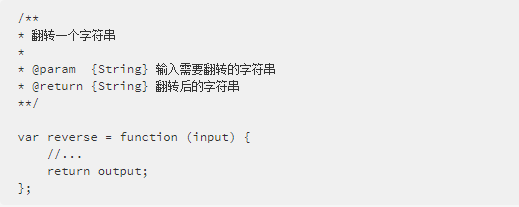


## 六、编写API文档

### 1. 生成API文档的步骤：

* 编写特殊格式的代码块（即一些注释块）
* 运行工具来解析代码和注释（工具如：JSDoc Toolkit和YUIDoc）
* 发布工具解析的结果，大多数情况是采用HTML格式发布（如网页版的API文档就是利用工具生成的）

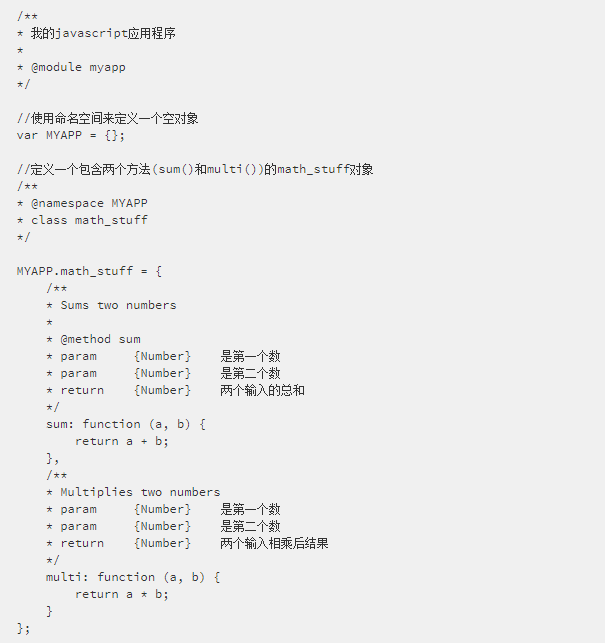
****简单举例****：



****YUIDoc范例****：

完整范例：本程序由一个文件(app.js)组成，该文件仅有一个模块(myapp)。

app.js:



@namespace：这里用于命名包含以上对象的全局引用的名称

@class：这里有些命名不当，他实际意思是指对象或者构造函数

@method：定义对象中的方法和方法名

@param：列举函数所使用的参数。其中将参数类型用大括号括起来，并在其后注释参数名及描述。

@return：类似于@param，这里用于描述返回值的，并且该方法没有名称。

@constructor：表明这个“类”实际上是一个构造函数

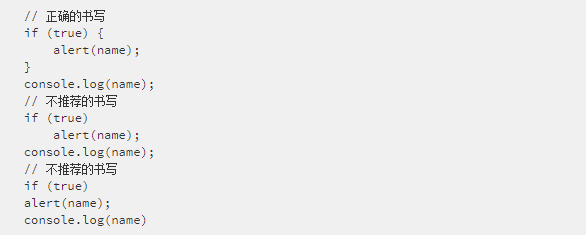
@property和@type描述了对象的属性。

### 2. 编写API目的：

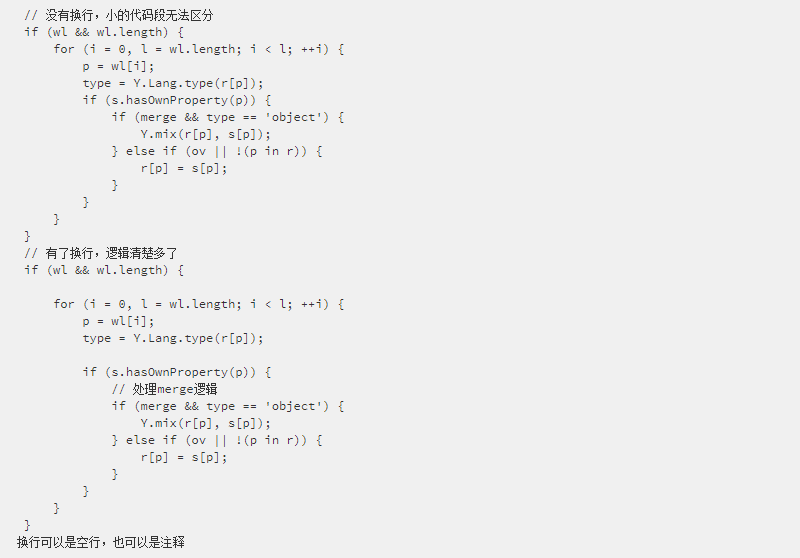
* 为API编写注释不仅仅是一中提供参考文档的简便方法，而且还有其他用途——通过再次审查代码，提高代码质量。
* 在解决问题时写出的解决方案仅仅是一个初稿。该解决方案可以给出令人期待的输出，但是该方案是否是最佳方案呢？改代码是否可读、易于理解、维护和升级呢？当您再次审视代码时您将更加确定代码哪些部分可以改进——如何使得代码更容易继续更新，移除一些不足之处等。它可以极大地帮助您创建高质量的代码。

## 七、推荐写法

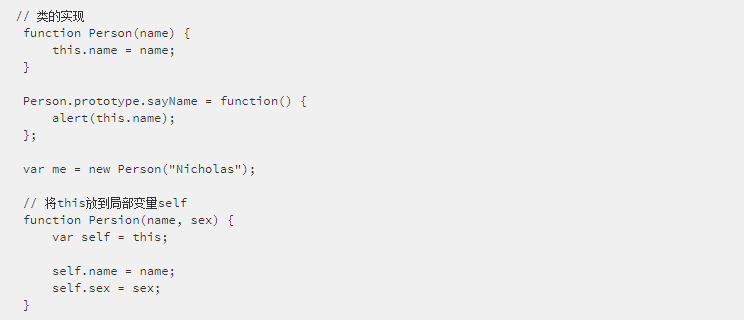
除了三目运算，if,else等禁止简写



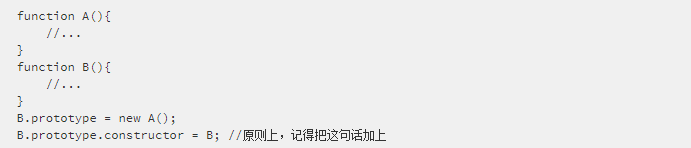
在需要以{}闭合的代码段前增加换行，如：for if



使用Function进行类的定义，不推荐继承，如需继承采用成熟的类库实现继承



平时咱们写代码，基本都是小程序，真心用不上什么继承，而且继承并不是JS的擅长的语言特性，尽量少用。如果非要使用的话，注意一点：



继承从原则上来讲，别改变他的构造函数，否则这个继承就显得很别扭了~

使用局部变量缓存反复查找的对象(包括但不限于全局变量、dom查询结果、作用域链较深的对象)



self是一个保留字，不过用它也没关系。在这里，看个人爱好吧，可以用\_this, that, me等这些词，都行，但是团队开发的时候统一下比较好。

使用eval，采取$.parseJSON

三个原因：

* 有注入风险，尤其是ajax返回数据
* 不方便debug
* 效率低，eval是一个执行效率很低的函数

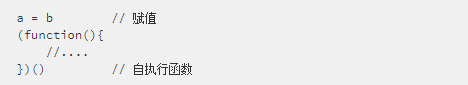
建议： 使用new Function来代替eval的使用，最好就别用。

## 八、不规范写法

句尾没有分号



这个是要引起注意的，比如：



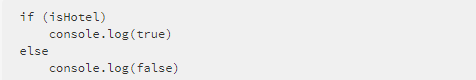
未加分号，结果被解析成



变量命名各种各样



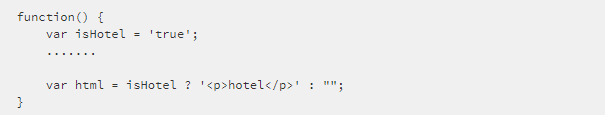
if 缩写



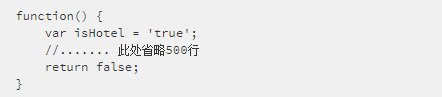
使用 eval



变量未定义到处都是



超长函数



## 九、需要注意的地方

window.onload只能使用一次，使用多次会被最后的覆盖。

解决方案：

* 只引用一次window.onload
* 使用jQuery的$(docuemnt.ready(function(){}); 可多次使用
* 使用函数封装，代码地址

CSS放在页头引入，javascript放在页尾引入

在上线之前，要编译压缩代码

减少重排与重绘