

# JavaScript代码风格

## 内容列表

1. 文件
2. 缩进
3. 换行
4. 分号
5. 引号
6. 逗号
7. 大括号
8. 中括号
9. 小括号
10. 对象属性
11. 变量声明
12. 函数
13. 注释

## 文件

- JavaScript 文件使用无 BOM 的 UTF-8 编码。

为什么？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，比如Mac系统中的[换行符问题](#)。

- JavaScript 文件结尾处留有一个换行符。

## 缩进

- 将 tab 设为 4 个空格。

```
// bad
function() {
  ..var name;
}

// bad
function() {
  .var name;
}

// good
function() {
  ....var name;
}
```

- if、else、switch、case、for、while 和 : 后1个空格。

```
// bad
if(a > b) {

}
// good
if (a > b) {

}

// bad
for(var i = 0; i < 5; i++) {
  // ...
}
// good
for (var i = 0; i < 5; i++) {
  // ...
}
```

5

- 赋值语句两边各 1 个空格

```
// bad
name='John';

// good
name = 'John';
```

- 两元运算符（`==`, `>`, `<`等）两边各 1 个空格。

```
// bad
if (a==b) {

}

// good
if (a == b) {

}
```

- 函数参数之间 1 个空格。

```
// bad
function ajax(url,param,success) {
    // ...
}

// good
function ajax(url, param, success) {
    // ...
}
```

- 左大括号前 1 个空格。

```
// bad
function ajax(url, param, success){
    // ...
}
if (a > b){

}

// good
function ajax(url, param, success) {
    // ...
}
if (a > b) {

}
```

- 语句行之间 4 个空格缩进，不使用Tab（可将编辑器的 `tab` 设置为 4 个空格）。

```
// bad
var name = 'John'; var obj = {};
if (name = 'Lily') { login(name); }

// good
var name = 'John';
var obj = {};
if (name = 'Lily') {
    login(name);
}
```

- 一元运算符（`++`, `--`）前后不加空格。

```
// bad
++ sum;
total --;
```

```
// good
++sum;
total--;
```

## 换行

- 每个独立语句结束后必须换行。

```
// bad
user.setCode(1); user.setName('admin'); user.setPwd('123456');
```

```
// good
user.setCode(1);
user.setName('admin');
user.setPwd('123456');
```

- 每行不得超过 120 个字符。超长的不可分割的代码允许例外，比如复杂的正则表达式。长字符串不在例外之列。

## 分号

- 语句始终使用分号结尾，`for`、`function`、`if`、`switch`、`try`、`while` 除外，不要依赖于引擎隐式插入。

```
// bad
function fn() {
    // ...
};
```

```
// good
function fn() {
    // ...
}
```

```
// bad
for (var i = 0; i < 5; i++) {
    // ...
};
```

```
// good
for (var i = 0; i < 5; i++) {
    // ...
}
```

## 逗号

- 不要将逗号放前面。

```
// bad
var hero = {
    firstName: 'Bob'
    , lastName: 'Parr'
    , heroName: 'Mr. Incredible'
    , superPower: 'strength'
};
```

```
// good
var hero = {
    firstName: 'Bob',
    lastName: 'Parr',
    heroName: 'Mr. Incredible',
    superPower: 'strength'
};
```

## 引号

- 引号多用来定义字符串，始终使用单引号。

```
// bad
var name = "John";
```

```
// good
var name = 'John';
```

```
// good
var name = 'John';
```

- HTML属性使用双引号，即单引号在外层，双引号在内层。

```
// bad
var html = "<a href='http://www.jd.com'>京东</a>";

// good
var html = '<a href="http://www.jd.com">京东</a>';
```

## 大括号

- 起首的大括号跟在关键字的后面，且大括号前留有一个空格。

```
// bad
if (a > b)
{
    // ...
}

// good
if (a > b) {
    // ...
}

// bad
function test(){
    console.log('test');
}

// good
function test() {
    console.log('test');
}

// bad
dog.set('attr',{
    age: '1 year',
    breed: 'Bernese Mountain Dog'
});

// good
dog.set('attr', {
    age: '1 year',
    breed: 'Bernese Mountain Dog'
});
```

## 中括号

- 定义数组，成员以逗号分隔，逗号后加 1 个空格。

```
// bad
var arr = [1,2,3,4];

// good
var arr = [1, 2, 3, 4];
```

- 取对象属性用点号，非标准属性才使用中括号。

```
var obj = {name: 'John'};

// bad
var name = obj['name'];

// good
var name = obj.name;
```

## 小括号

- 作为函数调用时函数名与左小括号之间没有空格。

```
// bad
func ();
```

```
// good
func();
```

- 作为强制运算符时左括号前加1个空格。

```
// bad
return(x + y);

// good
return (x + y);
```

## 对象属性

- 不加引号（除非一些特殊关键字或非合法标识符）。

```
// bad
var obj = {
  'name': 'John',
  'age': 30
}

// good
var obj = {
  name: 'John',
  age: 30
}
```

## 变量声明

- 驼峰式：变量、属性、方法名开始的第一个单词小写，之后的单词首字母大写。

```
// bad
var nickname = 'John';
var nick_name = 'John';

// good
var nickName = 'John';
```

- 避免单个字符名，变量名应具有描述意义（计数器除外）。

```
// bad
function q() {
  // ...stuff...
}

// good
function query() {
  // ..stuff..
}
```

- 常量名全部用大写字母。

```
// bad
var pi = 3.1415926;

// good
var PI = 3.1415926;
```

- 类（构造器）名首字母大写，用名词。

```
// bad
function person(name, age) {
  this.name = name;
  this.age = age;
}

// good
function Person(name, age) {
  this.name = name;
  this.age = age;
}
```

```
    var age = 30;  
}
```

- 多个变量，使用多个 `var` 关键字，不使用逗号分隔。

```
// bad  
var name = 'John',  
    age = 30,  
    gender = 'male';  
  
// good  
var name = 'John';  
var age = 30;  
var gender = 'male';
```

- 前缀参考

前缀	类型	示例
is, can has	Boolean	isPass
get	Getter	getName
set	Setter	setName
i, j, k	iterator	
el	HTMLElement	elNav
\$	jQuery Object	\$nav

## 函数

- 函数的长度控制在 50 行以内。

为什么？将过多的逻辑单元混在一个大函数中，难以维护。一个清晰易懂的函数应该完成单一的逻辑单元。复杂的操作应进一步分解，通过函数的调用来体现流程。

不可分割的特定算法逻辑允许例外。

- 函数的参数控制在 5 个以内，过多参数会导致维护难度增大。

为什么？有些函数的参数并非作为算法的输入，而是对算法的某些分支条件判断之用，此类参数建议通过一个JS对象 `options` 传递。

某些情况下，如使用 AMD/CMD Loader 的 `require` 加载多个模块时，其 `callback` 可能会存在较多参数，因此对函数参数的个数不做强制限制。

## 注释

- 在单行注释符后留一个空格。

```
// bad  
//this is comment  
  
// good  
// this is comment
```

- 在多行注释的结束符前留一个空格，使星号对齐。

```
// bad  
/*  
    this is comment  
*/  
  
// good  
/*  
    this is comment
```

```
*/
```

- 不要把注释写在多行注释的开始符、结束符所在行。

```
// bad
/* start

end */
/*
here is line 1
here is line 2
*/

// good
/*
    start
    end
*/
/*
    here is line 1
    here is line 2
*/
```

- 如果某段代码有功能未实现或有待完善，必须添加“TODO”标记，“TODO”前后应留一个空格。

```
// bad
// 未处理IE的兼容性
function setOpacity(node, val) {
    node.style.opacity = val;
}

// good
// TODO 未处理IE6-8的兼容性
function setOpacity(node, val) {
    node.style.opacity = val;
}
```

- 文档注释会以预定格式出现在API文档中。以“/”开头，“/”结束，其间的每一行均以“\*”开头且与开始符的第一个“\*”对齐，注释内容与“\*”间留一个空格。

```
// bad
/*
 * comment
*/

// good
/**
 * comment
*/
```

- 文档注释必须包含一个或多个注释标签。

```
// bad
/**
 * 模块说明
*/

// good
/**
 * 模块说明
 * @module 模块名
*/

/**
 * Core模块提供最基础的接口
 * @module Core
*/
```

- class 必须搭配@constructor或@static使用，分别标记非静态类与静态类。

```
// bad
/**
 * 组件基类
 * @class Component
 */
function Componetn(nodes) {
  // ...
}

// good
/**
 * 组件基类
 * @class Component
 * @constructor
 * @param {ArrayLike<Element>} nodes 初始化节点
 */
function Componetn(nodes) {
  // ...
}
```

- 声明函数或类方法，没有指定@for时，表示此函数为全局或顶层函数。当函数为静态函数时，必须添加@static；当函数有参数时，必须使用@param；当函数有返回值时，必须使用@return。

```
// bad
/**
 * 返回当前集合中指定位置的元素
 * @method
 */
Component.prototype.getNode = function(i) {
  // ...
}

// good
/**
 * 返回当前集合中指定位置的元素
 * @method
 * @for Component
 * @param {Number} 位置下标。如果为负数，则从集合的最后一个元素开始倒数
 * @return {Element} 指定元素
 */
Component.prototype.getNode = function(i) {
  // ...
}
```

- 声明对象属性，@property

```
// bad
/**
 * id 元素id
 * classNames 样式
 */
var htmlOptions = {
  id:null,
  classNames: null
}
htmlOptions.id = "123";
htmlOptions.classNames = "arrow area";

// good
/**
 * @property {IDString} id 元素id
 * @property {ClassString} classNames 样式
 */
var htmlOptions = {
  id:null,
  classNames: null
}
htmlOptions.id = "123";
```



```
htmlOptions.id = "123";  
htmlOptions.classNames = "arrow area";
```

- 文件注释位于文件的最前面，应包括文件的以下信息：概要说明及版本（必须）、项目地址（开源组件必须）、版权声明（必须）、开源协议（开源组件必须）、版本号（必须）、修改时间（必须）。

```
// bad  
/*!  
 * Z.js Javascript Library  
 */  
  
// good  
/*!  
 * Z.js Javascript Library v1.0.0  
 * @snandy - (2013-03-15)  
 * http://github.com/snandy/z.js  
 * Released under MIT license  
 */
```

[返回列表](#)