# JavaScript编码规范

[**1 前言**](https://github.com/fex-team/styleguide/blob/master/javascript.md#1-%E5%89%8D%E8%A8%80)

[**2 代码风格**](https://github.com/fex-team/styleguide/blob/master/javascript.md#2-%E4%BB%A3%E7%A0%81%E9%A3%8E%E6%A0%BC)

[**2.1 文件**](https://github.com/fex-team/styleguide/blob/master/javascript.md#21-%E6%96%87%E4%BB%B6)

[**2.2 结构**](https://github.com/fex-team/styleguide/blob/master/javascript.md#22-%E7%BB%93%E6%9E%84)

[**2.2.1 缩进**](https://github.com/fex-team/styleguide/blob/master/javascript.md#221-%E7%BC%A9%E8%BF%9B)

[**2.2.2 空格**](https://github.com/fex-team/styleguide/blob/master/javascript.md#222-%E7%A9%BA%E6%A0%BC)

[**2.2.3 换行**](https://github.com/fex-team/styleguide/blob/master/javascript.md#223-%E6%8D%A2%E8%A1%8C)

[**2.2.4 语句**](https://github.com/fex-team/styleguide/blob/master/javascript.md#224-%E8%AF%AD%E5%8F%A5)

[**2.3 命名**](https://github.com/fex-team/styleguide/blob/master/javascript.md#23-%E5%91%BD%E5%90%8D)

[**2.4 注释**](https://github.com/fex-team/styleguide/blob/master/javascript.md#24-%E6%B3%A8%E9%87%8A)

## 1 前言

JavaScript一直有着广泛的应用，特别是在浏览器端的行为管理。本文档的目标是使JavaScript代码风格保持一致，容易被理解和被维护。

虽然本文档是针对JavaScript设计的，但是在使用各种JavaScript的预编译语言时(如TypeScript等)时，适用的部分也应尽量遵循本文档的约定。

## 2 代码风格

### 2.1 文件

##### [建议] JavaScript 文件使用无 BOM 的 UTF-8 编码。

解释：

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。

##### [建议] 在文件结尾处，保留一个空行。

### 2.2 结构

#### 2.2.1 缩进

##### [强制] 一个Tab设置为4个空格键。换行缩进必须用Tab键。

示例：

function dataReload(){  
try{  
 if(!bindatainit){  
 documentReady();  
 }  
 return true;  
}catch(e){  
 return false;  
}

}

#### 2.2.2 空格

##### [强制] 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。

示例：

var a = !arr.length;

a++;

a = b + c;

##### [强制] 用作代码块起始的左花括号 { 前必须有一个空格。

示例：

// good

if (condition) {

}

while (condition) {

}

function funcName() {

}

// bad

if (condition){

}

while (condition){

}

function funcName(){

}

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

示例：

// good

if (condition) {

}

while (condition) {

}

(function () {

})();

// bad

if(condition) {

}

while(condition) {

}

(function() {

})();

##### [强制] 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。

示例：

// good

var obj = {

a: 1,

b: 2,

c: 3

};

// bad

var obj = {

a : 1,

b:2,

c :3

};

##### [强制] 行尾不得有多余的空格。

#### 2.2.3 换行

##### [强制] 每个独立语句结束后必须换行。

##### [强制] 每行不得超过 120 个字符。

解释：

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。

##### [强制] 运算符处换行时，运算符必须在新行的行首。

示例：

// good

if (user.isAuthenticated()

&& user.isInRole('admin')

&& user.hasAuthority('add-admin')

|| user.hasAuthority('delete-admin')

) {

// Code

}

var result = number1 + number2 + number3

+ number4 + number5;

// bad

if (user.isAuthenticated() &&

user.isInRole('admin') &&

user.hasAuthority('add-admin') ||

user.hasAuthority('delete-admin')) {

// Code

}

var result = number1 + number2 + number3 +

number4 + number5;

##### [建议] 不同行为或逻辑的语句集，使用空行隔开，更易阅读。

示例：

// 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;

}

##### [建议] 在语句的行长度超过 120 时，根据逻辑条件合理缩进。

示例：

// 较复杂的逻辑条件组合，将每个条件独立一行，逻辑运算符放置在行首进行分隔，或将部分逻辑按逻辑组合进行分隔。// 建议最终将右括号 ) 与左大括号 { 放在独立一行，保证与 if 内语句块能容易视觉辨识。

if (user.isAuthenticated()

&& user.isInRole('admin')

&& user.hasAuthority('add-admin')

|| user.hasAuthority('delete-admin')

) {

// Code

}

// 按一定长度截断字符串，并使用 + 运算符进行连接。// 分隔字符串尽量按语义进行，如不要在一个完整的名词中间断开。// 特别的，对于HTML片段的拼接，通过缩进，保持和HTML相同的结构。

var html = '' // 此处用一个空字符串，以便整个HTML片段都在新行严格对齐

+ '<article>'

+ '<h1>Title here</h1>'

+ '<p>This is a paragraph</p>'

+ '<footer>Complete</footer>'

+ '</article>';

// 也可使用数组来进行拼接，相对 + 更容易调整缩进。

var html = [

'<article>',

'<h1>Title here</h1>',

'<p>This is a paragraph</p>',

'<footer>Complete</footer>',

'</article>'

];

html = html.join('');

// 当参数过多时，将每个参数独立写在一行上，并将结束的右括号 ) 独立一行。// 所有参数必须增加一个缩进。

foo(

aVeryVeryLongArgument,

anotherVeryLongArgument,

callback

);

// 也可以按逻辑对参数进行组合。// 最经典的是baidu.format函数，调用时将参数分为“模板”和“数据”两块

baidu.format(

dateFormatTemplate,

year, month, date, hour, minute, second

);

// 当函数调用时，如果有一个或以上参数跨越多行，应当每一个参数独立一行。// 这通常出现在匿名函数或者对象初始化等作为参数时，如setTimeout函数等。setTimeout(

function () {

alert('hello');

},

200

);

order.data.read(

'id=' + me.model.id,

function (data) {

me.attchToModel(data.result);

callback();

},

300

);

// 链式调用较长时采用缩进进行调整。$('#items')

.find('.selected')

.highlight()

.end();

// 三元运算符由3部分组成，因此其换行应当根据每个部分的长度不同，形成不同的情况。

var result = thisIsAVeryVeryLongCondition

? resultA : resultB;

var result = condition

? thisIsAVeryVeryLongResult

: resultB;

// 数组和对象初始化的混用，严格按照每个对象的 { 和结束 } 在独立一行的风格书写。

var array = [

{

// ...

},

{

// ...

}

];

##### [建议] 对于 if...else...、try...catch...finally 等语句，推荐使用在 } 号后添加一个换行 的风格，使代码层次结构更清晰，阅读性更好。

示例：

if (condition) {

// some statements;

}else {

// some statements;

}

try {

// some statements;

}catch (ex) {

// some statements;

}

#### 2.2.4 语句

##### [强制] 不得省略语句结束的分号。

##### [强制] 在 if / else / for / do / while 语句中，即使只有一行，也不得省略块 {...}。

示例：

// good

if (condition) {

callFunc();

}

// bad

if (condition) callFunc();

### 2.3 命名

##### [强制] 常量 使用 全部字母大写，单词间下划线分隔 的命名方式。

示例：

var HTML\_ENTITY = {};

##### [强制]变量， 函数 ，函数的参数，类名使用 驼峰命名法。

示例：

function stringFormat(source) {

}

### 2.4 注释

所有js文件的开始部分都应有描述其功能的简要注释。这些注释并不描述细节信息（如何实现功能），这是因为细节有时要频繁更改。这样就可以避免不必要的注释维护工作以及错误的注释。细节信息由代码本身及必要的内部注释来描述。

每个重要的变量声明都应有内部注释，描述变量的用途。

应清楚地命名变量、控件和过程，仅在说明复杂细节时需要内部注释。

应在脚本的开始部分包含描述该脚本的概述，列举对象、过程、运算法则、对话框和其他系统从属物。有时一段描述运算法则的注释是很容易让人理解的。

注释一行代码用“//”标识，并且书写在需要注释的上一行，例如：

// 用于表示xx账户的名称  
var strAdName;

注释块用“/\*……/”标识，

示例：

/\*\*  
\*　主题：检查表单的完整性  
\*　作者：byron sampson  
\*　描述：检查提交表单时内容是否完整  
\*　修改日志  
\*　日期 修改原因 修改人 版本  
\*　201430131 创建 byron 1.1  
\*/

修改注释规范

为了管理好源代码，每一次修改文件都要进行详细记录，记录内容如例所示：

/\*\*  
\*　主题：检查表单的完整性  
\*　作者：byron sampson  
\*　描述：检查提交表单时内容是否完整  
\*　修改日志  
\*　日期 修改原因 修改人 版本  
\*　201430131 创建 byron 1.1  
\*　20150103 修改 byron 1.2  
\*/