**JS编码规范手册**

**XXXX科技股份有限公司**

**大数据研发中心**

**2017年7月**

**目录**

[1. 版本变更说明 3](#_Toc487805931)

[2. 概述 4](#_Toc487805932)

[为了提高工作效率，保证开发的有效性和合理性，并最大程度提高程序代码的可读性和可重复利用性，提高沟通效率，需要一份代码编写规范。 4](#_Toc487805933)

[3. 命名规则 5](#_Toc487805934)

[3.1 项目命名 5](#_Toc487805935)

[3.2目录命名 5](#_Toc487805936)

[3.2 JS文件命名 5](#_Toc487805937)

[4 JS代码书写规范 6](#_Toc487805938)

[4.1 缩进 6](#_Toc487805939)

[4.2单行长度 6](#_Toc487805940)

[4.3分号 6](#_Toc487805941)

[4.4空格 7](#_Toc487805942)

[4.5空行 8](#_Toc487805943)

[4.6换行 10](#_Toc487805944)

[4.7单行注释 11](#_Toc487805945)

[4.8多行注释 12](#_Toc487805946)

[4.9文档注释 12](#_Toc487805947)

[4.10引号 13](#_Toc487805948)

[4.11变量命名 13](#_Toc487805949)

[4.12变量声明 14](#_Toc487805950)

[4.13函数 14](#_Toc487805951)

[4.14数组、对象 15](#_Toc487805952)

[4.15 null 16](#_Toc487805953)

[4.16 undefined 16](#_Toc487805954)

[4.17 jshint 17](#_Toc487805955)

[4.18 杂项 19](#_Toc487805956)

# 版本变更说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 版本 | 日期 | 修改人 | 修改描述 |
| 1 | V1.0 | 2017-07-13 | xx | 创建 |

# 概述

为了提高工作效率，保证开发的有效性和合理性，并最大程度提高程序代码的可读性和可重复利用性，提高沟通效率，需要一份代码编写规范。

 本规范包含JS开发时程序编码中缩进，单行长度，分号，空格，空行，换行，单行注释，多行注释，文档注释，引号，变量命名，变量声明，函数，数组对象，括号等。

# 命名规则

## 项目命名

全部采用小写方式， 以下划线分隔。

例：my\_project\_name

## 3.2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, images, data\_models

## JS文件命名

参照项目命名规则。

例：account\_model.js

# JS代码书写规范

## 4.1 缩进

使用soft tab（4个空格）。

var x = 1,

y = 1;

if (x < y) {

x += 10;

} else {

x += 1;

}

## 4.2单行长度

不要超过80，但如果编辑器开启word wrap可以不考虑单行长度。

## 4.3分号

以下几种情况后需加分号：

变量声明

表达式

return

throw

break

continue

do-while

/\* var declaration \*/

var x = 1;

/\* expression statement \*/

x++;

/\* do-while \*/

do {

x++;

} while (x < 10);

## 4.4空格

以下几种情况不需要空格：

对象的属性名后

前缀一元运算符后

后缀一元运算符前

函数调用括号前

无论是函数声明还是函数表达式，'('前不要空格

数组的'['后和']'前

对象的'{'后和'}'前

运算符'('后和')'前

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++;

}

## 4.5空行

以下几种情况需要空行：

变量声明后（当变量声明在代码块的最后一行时，则无需空行）

注释前（当注释在代码块的第一行时，则无需空行）

代码块后（在函数调用、数组、对象中则无需空行）

文件最后保留一个空行

// need blank line after variable declaration

var x = 1;

// not need blank line when variable declaration is last expression in the current block

if (x >= 1) {

var y = x + 1;

}

var a = 2;

// need blank line before line comment

a++;

function b() {

// not need blank line when comment is first line of block

return a;

}

// need blank line after blocks

for (var i = 0; i < 2; i++) {

if (true) {

return false;

}

continue;

}

var obj = {

foo: function() {

return 1;

},

bar: function() {

return 2;

}

};

// not need blank line when in argument list, array, object

func(

2,

function() {

a++;

},

3

);

var foo = [

2,

function() {

a++;

},

3

];

var foo = {

a: 2,

b: function() {

a++;

},

c: 3

};

## 4.6换行

换行的地方，行末必须有','或者运算符；

以下几种情况不需要换行：

下列关键字后：else, catch, finally

代码块'{'前

以下几种情况需要换行：

代码块'{'后和'}'前

变量赋值后

// not good

var a = {

b: 1

, c: 2

};

x = y

? 1 : 2;

// good

var a = {

b: 1,

c: 2

};

x = y ? 1 : 2;

x = y ?

1 : 2;

// no need line break with 'else', 'catch', 'finally'

if (condition) {

...

} else {

...

}

try {

...

} catch (e) {

...

} finally {

...

}

// not good

function test()

{

...

}

// good

function test() {

...

}

// not good

var a, foo = 7, b,

c, bar = 8;

// good

var a,

foo = 7,

b, c, bar = 8;

## 4.7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。

if (condition) {

// if you made it here, then all security checks passed

allowed();

}

var zhangsan = 'zhangsan'; // one space after code

## 4.8多行注释

最少三行, '\*'后跟一个空格，具体参照右边的写法；

建议在以下情况下使用：

难于理解的代码段

可能存在错误的代码段

浏览器特殊的HACK代码

业务逻辑强相关的代码

/\*

\* one space after '\*'

\*/

var x = 1;

## 4.9文档注释

各类标签@param, @method等

建议在以下情况下使用：

所有常量

所有函数

所有类

/\*\*

\* @func

\* @desc 一个带参数的函数

\* @param {string} a - 参数a

\* @param {number} b=1 - 参数b默认值为1

\* @param {string} c=1 - 参数c有两种支持的取值</br>1—表示x</br>2—表示xx

\* @param {object} d - 参数d为一个对象

\* @param {string} d.e - 参数d的e属性

\* @param {string} d.f - 参数d的f属性

\* @param {object[]} g - 参数g为一个对象数组

\* @param {string} g.h - 参数g数组中一项的h属性

\* @param {string} g.i - 参数g数组中一项的i属性

\* @param {string} [j] - 参数j是一个可选参数

\*/

function foo(a, b, c, d, g, j) {

...

}

## 4.10引号

最外层统一使用单引号。

// not good

var x = "test";

// good

var y = 'foo',

z = '<div id="test"></div>';

## 4.11变量命名

标准变量采用驼峰式命名

'ID'在变量名中全大写

'URL'在变量名中全大写

'Android'在变量名中大写第一个字母

'iOS'在变量名中小写第一个，大写后两个字母

常量全大写，用下划线连接

构造函数，大写第一个字母

jquery对象必须以'$'开头命名

var thisIsMyName;

var goodID;

var reportURL;

var AndroidVersion;

var iOSVersion;

var MAX\_COUNT = 10;

function Person(name) {

this.name = name;

}

// not good

var body = $('body');

// good

var $body = $('body');

## 4.12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;

}

}

## 4.13函数

无论是函数声明还是函数表达式，'('前不要空格，但'{'前一定要有空格；

函数调用括号前不需要空格；

立即执行函数外必须包一层括号；

不要给inline function命名；

参数之间用', '分隔，注意逗号后有一个空格。

// no space before '(', but one space before'{'

var doSomething = function(item) {

// do something

};

function doSomething(item) {

// do something

}

// not good

doSomething (item);

// good

doSomething(item);

// requires parentheses around immediately invoked function expressions

(function() {

return 1;

})();

// not good

[1, 2].forEach(function x() {

...

});

// good

[1, 2].forEach(function() {

...

});

// not good

var a = [1, 2, function a() {

...

}];

// good

var a = [1, 2, function() {

...

}];

// use ', ' between function parameters

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

// do something

};

## 4.14数组、对象

对象属性名不需要加引号；

对象以缩进的形式书写，不要写在一行；

数组、对象最后不要有逗号。

// not good

var a = {

'b': 1

};

var a = {b: 1};

var a = {

b: 1,

c: 2,

};

// good

var a = {

b: 1,

c: 2

};

## 4.15 null

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

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) {

...

}

## 4.16 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') {

...

}

## 4.17 jshint

用'===', '!=='代替'==', '!='；

for-in里一定要有hasOwnProperty的判断；

不要在内置对象的原型上添加方法，如Array, Date；

不要在内层作用域的代码里声明了变量，之后却访问到了外层作用域的同名变量；

变量不要先使用后声明；

不要在一句代码中单单使用构造函数，记得将其赋值给某个变量；

不要在同个作用域下声明同名变量；

不要在一些不需要的地方加括号，例：delete(a.b)；

不要使用未声明的变量（全局变量需要加到.jshintrc文件的globals属性里面）；

不要声明了变量却不使用；

不要在应该做比较的地方做赋值；

debugger不要出现在提交的代码里；

数组中不要存在空元素；

不要在循环内部声明函数；

不要像这样使用构造函数，例：new function () { ... }, new Object；

// not good

if (a == 1) {

a++;

}

// good

if (a === 1) {

a++;

}

// good

for (key in obj) {

if (obj.hasOwnProperty(key)) {

// be sure that obj[key] belongs to the object and was not inherited

console.log(obj[key]);

}

}

// not good

Array.prototype.count = function(value) {

return 4;

};

// not good

var x = 1;

function test() {

if (true) {

var x = 0;

}

x += 1;

}

// not good

function test() {

console.log(x);

var x = 1;

}

// not good

new Person();

// good

var person = new Person();

// not good

delete(obj.attr);

// good

delete obj.attr;

// not good

if (a = 10) {

a++;

}

// not good

var a = [1, , , 2, 3];

// not good

var nums = [];

for (var i = 0; i < 10; i++) {

(function(i) {

nums[i] = function(j) {

return i + j;

};

}(i));

}

// not good

var singleton = new function() {

var privateVar;

this.publicMethod = function() {

privateVar = 1;

};

this.publicMethod2 = function() {

privateVar = 2;

};

};

## 4.18 杂项

不要混用tab和space；

不要在一处使用多个tab或space；

换行符统一用'LF'；

对上下文this的引用只能使用'\_this', 'that', 'self'其中一个来命名；

行尾不要有空白字符；

switch的falling through和no default的情况一定要有注释特别说明；

不允许有空的代码块。

// not good

var a = 1;

function Person() {

// not good

var me = this;

// good

var \_this = this;

// good

var that = this;

// good

var self = this;

}

// good

switch (condition) {

case 1:

case 2:

...

break;

case 3:

...

// why fall through

case 4

...

break;

// why no default

}

// not good with empty block

if (condition) {

}