|  |
| --- |
|  |
| JavaScript规范 v1.2 |
|  |

|  |
| --- |
| 大电商产品开发 |

将基于陈迪 方彬 20130923版本进行补充与修改。

目录

[1. 背景 1](#_Toc367867498)

[2. JavaScript文件 1](#_Toc367867499)

[3. 缩进 1](#_Toc367867500)

[4. 行长度 1](#_Toc367867501)

[5. 注释 1](#_Toc367867502)

[6. 变量声明 2](#_Toc367867503)

[7. 函数声明 2](#_Toc367867504)

[8. 命名 3](#_Toc367867505)

[9. 语句 4](#_Toc367867506)

[简单语句 4](#_Toc367867507)

[复合语句 4](#_Toc367867508)

[return语句 5](#_Toc367867509)

[if 语句 5](#_Toc367867510)

[for语句 5](#_Toc367867511)

[while语句 6](#_Toc367867512)

[do语句 6](#_Toc367867513)

[switch语句 6](#_Toc367867514)

[try语句 7](#_Toc367867515)

[continue语句 7](#_Toc367867516)

[with语句 7](#_Toc367867517)

[空白 7](#_Toc367867518)

[10. 建议 8](#_Toc367867519)

[{} 和 [] 8](#_Toc367867520)

[不允许修改内建对象的prototype 8](#_Toc367867521)

[使用prototype扩展已有的类 8](#_Toc367867522)

[,（逗号）操作符 8](#_Toc367867523)

[块范围 9](#_Toc367867524)

[赋值表达式 9](#_Toc367867525)

[===和!== 9](#_Toc367867526)

[令人迷惑的+和- 9](#_Toc367867527)

[eval是及其糟糕的！ 9](#_Toc367867528)

[避免JavaScript中的内存泄露问题 10](#_Toc367867529)

## 背景

这是一份针对JavaScript编程的约定和规则。此文档受Sun的《Java编程语言的编码规范》的启迪，当然是大大改动过的。

对于组织来说，软件的长期质量直接正比于代码库的质量。在一个程序的生命周期中，它会经手很多程序员。如果程序本身能清楚地传达它的结构和特性，那么在将来被修改时，它被彻底搞砸的几率也会变小。

我们写所有的JavaScript代码都是直接传送给公众的，所以应该有出版一般的质量。

整洁的代码是重要的。

## JavaScript文件

1. JavaScript文件以外链形式 .js 格式文件保存和传输。
2. 不要把JavaScript代码嵌在HTML文件里面（除非特殊需求或需要增强用户体验）。
3. <script src=“”></script>形式的外链JS文件放在HTML页面结构的底部。没有必要指定<script>标签的language和type属性。
4. 用Lab.js方法加载相应的外链JS文件。

## 缩进

1. 以4个空格为缩进的单位。可以将tab调整成4个空格来使用tab缩进。

## 行长度

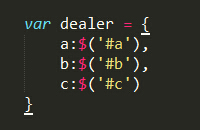
1. 避免一行超过80个字符。当一条语句在一行放不下的时候，建议断行。将断行放在操作符（如逗号）之后，这样可以避免在复制粘贴代码后后插入分号的错误。换行后的下一行应该缩进8个字符。

## 注释

1. 养成良好的注释习惯。注释应当清晰，有意义，告诉阅读者你都干了些什么。
2. 代码变动后要及时更新注释。错误的注释会导致程序更加难以阅读和理解。
3. JavaScript有行注释和块注释两种。一般建议使用行注释，块注释用来做正式的文档说明、模块说明或注释掉代码块。

## 变量声明

1. 在函数的头部定义所有的变量。var语句应该是函数体的第一条语句。
2. 尽量少使用全局变量。永远不要使用隐式的全局变量。
3. **如果全局变量太多，用对象字面量，以减少全局变量的影响及风险，如下图：**

****

**Dealer为你所开发版块的名字。**

1. **常量定义必须全部用大写字母，如：NAMES\_LIKE\_THIS**
2. **使用单引号来定义字符串，如图：**

**C:\Users\wangjingliang\AppData\Roaming\Tencent\Users\546172171\QQ\WinTemp\RichOle\9Y6S(@OULW7PC8`A3{V558L.jpg**

1. **私有成员变量或者方法名字以下划线开头，如 var \_this=**
2. **变量命名样式比较多，如下：**

**var is\_name;**

**var isName;**

**var isname;**

**我们建议用中间的，驼峰形式；**

1. **不要将目的不同的语句，合并成一行书写，很容易让人误解，如下图：**
2. C:\Users\wangjingliang\AppData\Roaming\Tencent\Users\546172171\QQ\WinTemp\RichOle\R78(F~LUOYEOK5[J)}M8T28.jpg

**你的意思也许是这样：**

**b=0;**

**var a=b;**

**而好多人会理解为：a=0;b=0;**

## 函数声明

1. 所有的函数在使用之前声明。内部函数应该在var语句之后，这样清楚显示哪些变量使用在内部函数范围内。
2. A约定在JavaScript中很有用。在JavaScript中，函数和字面量对象可以被放置在任何表达式能放置的位置。这样约定对内联函数和复杂结构有最好的可读性。

functiongetElementsByClassName(className) {

var results = [];

walkTheDOM(document.body, function (node) {

var a; // array of class names

var c = node.className; // the node's classname

vari; // loop counter

// If the node has a class name, then split it into a list of simple names.

// If any of them match the requested name, then append the node to the set of results.

if (c) {

a = c.split(' ');

for (i = 0; i<a.length; i += 1) {

if (a[i] === className) {

results.push(node);

break;

}

}

}

});

return results;

}

1. 匿名函数应该在“function”和左括号之间有一个空格。如果空格省略了，看上就好像函数名是function。

div.onclick = function (e) {

return false;

};

that = {

method: function () {

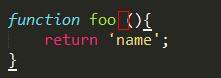
return this.datum;

},

datum: 0

};

1. 全局函数应该越少用越好。
2. 如果是IIF（立即执行函数），那么整个表达式应该用括号包起来。这样，能清楚地表明返回值是函数执行结果而不是函数本身。
3. **普通函数以小写开头驼峰形式命名。构造函数以大写字母开头的驼峰式命名；**
4. **函数后不添加空格。（据统计68%的人没有空格）**



**H． 参数之间，参数与括号之间不添加任何空格。**

## 命名

1. 命名由26个大小写英文字母，数字和下划线\_组成。避免使用国际字符。避免使用$美元符号和\斜杠。
2. 第一个字母避免使用下划线\_。如果表示私有，请使用那些能够提供私有成员的形式。不要用看上去具有私有能力但实际没有的名字命名。
3. 大部分情况下建议变量和函数应该以小写字符开头。
4. 必须以new前缀来使用的构造函数，应该用大写字母开头。如果new被省略了，JavaScript不会提供编译时警告或者运行时警告，但不使用new后果很严重，首字母大写是我们唯一的防御手段。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 原则 | 正确举例 | 错误举例 |
| 类名 | 一个或多个单词构成，每个单词的首字母大写，其余字母小写，单词之间直接相连，没有其他符号。 | TestClass | Test\_Class、testClass、test\_class等 |
| 公共方法 | 一个或多个单词构成，第一个单词的所有字母均小写，其余每个单词的首字母大写，其余字母小写，单词之间直接相连，没有其他符号。 | testMethod(), getHtml(), getXml() | TestMethod(), test\_Method(), test\_method(), getHTML(), getXML() |
| 全局变量 | 以“g\_”开头，而后由一个或多个单词构成，第一个单词的所有字母均小写，其余每个单词的首字母大写，其余字母小写，单词之间直接相连，没有其他符号。这里，第一个单词前，可使用一个缩写短语表示全局变量的类型，如果使用了类型缩写短语，那么其余所有单词的首字母均大写，其余字母小写。这里的变量类型缩写包括：  n 表示某种节点（如XML节点、HTML节点等），如nText等  i 表示整数的列举形式，如nValue等  lst list类型  s string类型  d Date类型  f 有小数点的数字  b 布尔类型 | g\_fileName, g\_lstMenuItems | testVar, TestVar, test\_var |
| 常量 | 一个或多个单词构成，每个单词的所有字母均大写，单词之间用“\_”相连。 | TEST\_CONST | test\_const, TESTCONST |

## 语句

### 简单语句

1. 一条语句一行，必须以分号结束。
2. JavaScript允许表达式用作语句。这可能掩盖一些错误，特别是在有分号插入的时候。只有赋值和函数调用这两个表达式才能用作语句。

### 复合语句

1. 复合语句是指用大括号括起的多条语句而形成的语句。
2. 被括起的语句缩进4个空格。
3. 左大括号应该在开始复合语句的那行的末尾。
4. 右大括号应该在一行的开头，和复合语句开始的那行的开头对齐。
5. 控制结构(for, if等)上的语句必须使用大括号，即便其内部只有一条语句。这样比较容易添加语句而不必担心意外地引入bug。

### return语句

1. 返回值的return语句不应该使用()。返回值表达式应该和return语句同一行，以避免分号插入。

### if 语句

1. if语句应该具有如下形式

if (condition) {

statements

}

if (condition) {

statements

} else {

statements

}

if (condition) {

statements

} else if (condition) {

statements

} else {

statements

}

### for语句

* 1. for语句应该有如下的形式：

for (initialization; condition; update) {

statements

}

for (variable in object) {

if (filter) {

statements

}

}

* 1. 第一种应该用到数组上和那些可预先确定迭代次数的循环上。
  2. 第二种形式应该用在对象上。注意加在对象原型上的成员也会出现在枚举中。编程时，可以用hasOwnProperty方法来区分对象真正的对象成员和对象原型成员。

for (variable in object) {

if (object.hasOwnProperty(variable)) {

statements

}

}

### while语句

1. 一个while语句应该有如下的形式

while (condition) {

statements

}

### do语句

1. 一条do语句应该有如下形式

do {

statements

} while (condition);

1. 不像其他符合语句，do语句始终以；分号结尾。

### switch语句

1. 一条switch语句应该有如下形式

switch (expression) {

case expression:

statements

default:

statements

}

1. 每个case都应该和switch对齐，以避免过度缩进。
2. 每组语句都应该以break, return和throw结尾。不要一个case贯穿多个case。

### try语句

1. try系列语句应该有如下形式

### continue语句

1. 避免使用continue语句。Continue有模糊函数执行过程(control flow)。

try {

statements

} catch (variable) {

statements

}

try {

statements

} catch (variable) {

statements

} finally {

statements

}

### with语句

1. With语句不应该使用。

### 空白

1. 用空格将逻辑上相关的代码分为一节，以增加可阅读性。
2. 空格应该在下面这些情况下使用：
3. 关键字后跟左括号，他们之间应该有一个空格。
4. 所有的二元操作符，除了. ( [应该和他们的操作数之间用一个空格分开。
5. 一元操作符和它的操作数之间不应该加空格，除了typeof。
6. for控制部分中，每个分号;后面要跟一个空格。
7. 每个逗号，后面都要加空格。

## 建议

**不要出现多行的字符串常量**

//可以

Var TEST\_DATA = “AAAAAAAAAAAA”

+”BBBBBBBBBBBBB”

+”CCCCCCCCCCCCC”;

//不要

Var TEST\_DATA = “AAAAAAAAAAAA\

BBBBBBBBBBBBB\

CCCCCCCCCCCCC”; //”\”为分行符

### {} 和 []

1. 不要用new Object()，用{}。用[]而不要用new Array()。
2. 当成员名字是顺序整数的时候，使用数组；当成员名字是任意的字符串或名字时，使用对象。

### 不允许修改内建对象的prototype

1. 不要修改Object和Array。

### 使用prototype扩展已有的类

1. Number.add(num)：作用，数字相加
2. 实现方法：Number.prototype.add = function(num){return(this+num);}
3. 试验：alert((3).add(15)) -> 显示 18

### ,（逗号）操作符

1. 避免使用逗号操作符，除了在for语句的控制部分可谨慎使用。（这条规则不针对逗号分隔符，例如在字面量对象，字面量数组，var语句和参数列表）。

### 块范围

1. 在JavaScript中，块没有作用范围。只有函数有作用范围。不要使用块，除了在复合语句中使用的块。

### 赋值表达式

1. 避免在if和while语句的条件判断部分使用赋值语句。

if (a = b) {

if (a == b) {

哪个才是你想要表达的?

### ===和!==

1. ===和!==几乎总是比==和!==好的。==和!=操作符会有类型的强制转换。特别是不要用==来和假值（falsy values）进行比较。
2. （falsy values在JS中是：false, 0, “”, null, undefined, NaN，具体来说，false、0和””使用==操作符的比较结果是true，而null和undefined除了和自己和对方相等以外，和任何对象都不相等，NaN连自己都不相等！）

### 令人迷惑的+和-

1. 小心不要在一个+加号后面跟加号+或双加号++。这种模式令人困惑。在他们中间加上括号使你的意图清晰。

total = subtotal + +myInput.value;

最好写成下面这样

total = subtotal + (+myInput.value);

这样+ +不会被误认为++

### eval是及其糟糕的！

Eval函数在JS中是最被误用的特性。避免使用！

Eval有别名，不要使用Function构造函数，不要给setTimeout和setInterval传递字符串。

**能传递function时候不要传递string**

function dolaert() {

alert("OK");

}

setTimeout("dolaert()"); //不建议

setTimeout(dolaert); //建议

setTimeout(function dolaert(){});//建议

### 避免JavaScript中的内存泄露问题

待讨论的问题。