JavaScript 程序编码规范

中文版：http://yeeyan.com/articles/view/cloudwater/4042
英文版：http://javascript.crockford.com/code.html
译者：cloudwater

整理：手气不错 <i.feelinglucky@gmail.com>
主页：http://www.gracecode.com

整理日期：2008 年 01 月 22 日

前言

Any violation to this guide is allowed if it enhances readability.

所有的代码都要变成可供他人容易阅读的。

软件的长期价值直接源于其编码质量。在它的整个生命周期里，一个程序可能会被许多人阅读或修改。如果一个程序可以清晰的展现出它的结构和特征，那就能减少在以后对其进行修改时出错的可能性。

编程规范可以帮助程序员们增加程序的健壮性。所有的 JavaScript 代码都是暴露给公众的。所以我们更应该保证其质量。保持整洁很重要。

JavaScript 文件

JavaScript 程序应独立保存在后缀名为 .js 的文件中。

JavaScript 代码不应该被包含在 HTML 文件中，除非这是段特定只属于此部分的代码。在 HTML 中的JavaScript代码会明显增加文件大小，而且也不能对其进行缓存和压缩。

filename.js 应尽量放到 body 的后面。这样可以减少因为载入脚本而造成其他页面内容载入也被延迟的问题。也没有必要使用 language 或者 type 属性。MIME 类型是由服务器而非 scripttag 来决定的。（手气不错：个人认为按照 Web 标准而言，建议指定 type 属性，并将 <script> 放到页面的 <head> 中。）

缩进

缩进的单位为四个空格。避免使用 Tab 键来缩进。因为始终没有个统一的 Tab 长短标准。虽然使用空格会增加文件的大小，但在局域网中几乎可以忽略，且在最小化过程中也可被消除掉。

每行长度

避免每行超过 80 个字符。当一条语句一行写不下时，请考虑折行。在运算符号，最好是逗号后换行。在运算符后换行可以减少因为复制粘贴产生的错误被分号掩盖的几率。

注释

不要吝啬注释。给以后需要理解你的代码的人们（或许就是你自己）留下信息是非常有用的。注释应该和它们所注释的代码一样是书写良好且清晰明了。偶尔的小幽默就更不错了。记得要避免冗长或者情绪化。

及时地更新注释也很重要。错误的注释会让程序更加难以阅读和理解。

让注释有意义。重点在解释那些不容易立即明白的逻辑上。不要把读者的时间浪费在阅读类似于:

```
i = 0; // 让 i 等于 0
```

使用单行注释。块注释用于注释正式文档和无用代码。

变量声明

所有的变量必须在使用前进行声明。JavaScript 并不强制必须这么做，但这么做可以让程序易于阅读，且也容易发现那些没声明的变量(它们会被编译成*全局变量*)。

将 var 语句放在函数的首部。

最好把每个变量的声明语句单独放到一行，并加上注释说明。所有变量按照字母排序。

```
var currentEntry; // 当前选择项
var level;        // 缩进程度
var size;         // 表格大小
```

JavaScript 没有块范围，所以在块里面定义变量很容易引起 C/C++/Java 程序员们的误解。在函数的首部定义所有的变量。

*尽量减少*全局变量的使用。不要让局部变量覆盖全局变量。

函数声明

所有的函数在使用前进行声明。 内函数的声明跟在 var 语句的后面。这样可以帮助判断哪些变量是在函数范围内的。

函数名与“(”（左括号）之间不应该有空格。“)”（右括号）与开始程序体的“{”（左大括号）之间应插入一个空格。函数程序体应缩进四个空格。“}”（右大括号）与声明函数的那一行代码头部对齐。

```
function outer(c, d) {
    var e = c * d;

    function inner(a,  b) {
        return (e * a) + b;
    }

    return inner(0,  1);
}
```

下面这种书写方式可以在 JavaScript 中正常使用，因为在 JavaScript 中，函数和对象的声明可以放到任何表达式允许的地方。且它让内联函数和混合结构具有最好的可读性。

```
function getElementsByClassName(className) {
    var results = [];
    walkTheDOM(document.body, function (node) {
        var a;                  // array of class names
```

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

如果函数是匿名函数，则在 function 和 “(”（左括号）之间应有一个空格。如果省略了空格，否则会让人感觉函数名叫作 function。

```
    div.onclick = function (e) {
        return false;
    };

    that = {
        method: function () {
            return this.datum;
        },
        datum: 0
    };
```

尽量不使用全局函数。

命名

变量名应由 26 个大小写字母（A..Z，a..z），10 个数字（0..9），和 “_”（下划线）组成。避免使用国际化字符（如中文），因为它们不是在任何地方都可以被方便的阅读和理解。不要在命名中使用 “$”（美元符号）或者 “\”（反斜杠）。

不要把 “_”（下划线）作为变量名的第一个字符。它有时用来表示私有变量，但实际上 JavaScript 并没提供私有变量的功能。如果私有变量很重要，那么使用 *私有成员* 的形式。应避免使用这种容易让人误解的命名习惯。

大多数的变量名和方法命应以小写字母开头。

必须与 new 共同使用的构造函数名应以大写字母开头。当 new 被省略时 JavaScript 不会有任何编译错误或运行错误抛出。忘记加 new 时会让不好的事情发生（比如被当成一般的函数），所以大写构造函数名是我们来尽量避免这种情况发生的唯一办法。

全局变量应该全部大写。（JavaScript 没有宏或者常量，所以不会因此造成误会）

语句

简单语句

每一行最多只包含一条语句。把 “;” （分号）放到每条简单语句的结尾处。注意一个函数赋值或对象赋值语句也是赋值语句，应该以分号结尾。

JavaScript 可以把任何表达式当作一条语句。这很容易隐藏一些错误，特别是误加分号的错误。只有在赋值和调用时，表达式才应被当作一条单独的语句。

复合语句

复合语句是被包含在 “{ }”（大括号）的语句序列。

被括起的语句必须多缩进四个空格。

“{”（左大括号）应在复合语句其实行的结尾处。

“}”（右大括号）应与“{”（左大括号）的那一行的开头对齐。

大括号应该在所有复合语句中使用，即使只有一条语句，当它们是控制结构的一部分时，比如一个 if 或者 for 语句。这样做可以避免以后添加语句时造成的错误。

标示

语句标示是可选的，只有以下语句必须被标示：while、do、for、switch。

return 语句

一条有返回值的 return 语句不要使用“( )”（括号）来括住返回值。如果返回表达式，则表达式应与 return 关键字在同一行，以避免误加分号错误。

if 语句

if 语句应如以下格式:

```
if (condition){
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

for 语句

for 语句应如以下格式:

```
for (initialization;condition; update) {
    statements;
}

for (variable in object) if (filter) {
    statements;
}
```

第一种形式的循环用于已经知道相关参数的数组循环。

第二种形式应用于对象中。Object 原型中的成员将会被包含在迭代器中。通过预先定义 hasOwnProperty 方法来区分真正的 object 成员是个不错方法：

```
for (variablein object) if (object.hasOwnProperty(variable)){
    statements;
}
```

while 语句

while 语句应如以下格式:

```
while (condition){
    statements;
}
```

do 语句

do 语句应如以下格式:

```
do {
    statements;
} while (condition);
```

不像别的复合语句，do 语句总是以 “;”（分号）结尾。

switch 语句

switch 语句应如以下格式:

```
switch (expression){
case expression:
    statements;
default:
    statements;
}
```

每个 case 与 switch 对齐。这可避免过分缩进。

每一组 statements（除了default 应以 break，return，或者 throw 结尾），不要让它顺次往下执行。

try 语句

try 语句应如以下格式:

```
try {
    statements;
} catch (variable){
    statements;
}

try {
    statements;
} catch (variable){
    statements;
} finally {
```

```
        statements;
    }
```

continue 语句

*避免*使用 continue 语句。它很容易使得程序的逻辑过程晦涩难懂。

with 语句

不要使用 with 语句。

空白

用空行来将逻辑相关的代码块分割开可以提高程序的可读性。空格应在以下情况时使用：

跟在“(”（左括号）后面的关键字应被一个空格隔开。

```
    while (true) {
```

函数参数与“(”（左括号）之间不应该有空格。这能帮助区分关键字和函数调用。所有的二元操作符，除了“.”（点） 和“(”（左括号）和 “[”（左方括号）应用空格将其与操作数隔开。

一元操作符与其操作数之间不应有空格，除非操作符是个单词，比如 typeof。

每个在控制部分，比如 for 语句中的 “;”（分号）后须跟一个空格。

每个 “,”（逗号）后应跟一个空格。

另外的建议

{} 和[]

使用 `{}` 代替 `new Object()`。使用 `[]` 代替 `new Array()` 。

当成员名是一组有序的数字时使用数组来保存数据。当成员名是无规律的字符串或其他时使用对象来保存数据。

,（逗号）操作符

避免使用逗号操作符，除非在特定的 for 语句的控制部分。（这不包括那些被用在对象定义，数组定义，var 语句，和参数列表中的逗号分隔符。）

作用域

在 JavaScript 中块没有域。只有函数有域。不要使用块，除非在复合语句中。

赋值表达式

避免在 if 和 while 语句的条件部分进行赋值。

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

是一条正确语句？或者

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

才是对的？避免这种不容易判断对错的结构。

=== 和 !== 操作符。

使用 === 和 !== 操作符会相对好点。== 和 != 操作符会进行类型强制转换。 特别是， 不要将 == 用于与错值比较（false，null，undefined，“”，0，NaN）。

令人迷惑的加号和减号

小心在 + 后紧跟 + 或 ++。这种形式很容易仍人迷惑。应插入括号以便于理解。

```
total = subtotal + +myInput.value;
```

最好能写成

```
total = subtotal + (+myInput.value);
```

这样 ++ 不会被误认为是 ++。

eval 是恶魔

eval 是 JavaScript 中最容易被滥用的方法。避免使用它。

eval 有别名。不要使用 function 构造器。不要给 setTimeout 或者 setInterval 传递字符串参数。

后记

还有一份 Dojo 的编程规范可用作对比参考：http://dojotoolkit.org/developer/StyleGuide（中文版：http://code.google.com/p/grace/wiki/DojoStyle）。