Javascript规范

JavaScript灵活，简单易懂，对代码的格式的要求也相对松散。也正因为这样，JavaScript 的编码规范也往往被轻视，开发过程中修修补补，最终也就演变成为后续维护人员的噩梦。另外， JavaScript 代码是直接发送给客户浏览器的，直接与客户见面，编码的质量影响用户对软件的直接感受。

1. **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。

<script src=filename.js>应尽量放到body的后面。这样可以减少因为载入script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。

**2、缩进**

缩进的单位为四个空格。避免使用Tab键来缩进(即使现在已经是21世纪了),也始终没有个统一的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。

**3、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。下一行应该缩进8个空格。

**4、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。

**5、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,但是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](http://yuiblog.com/blog/2006/06/01/global-domination/))。

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。

**6、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; // 类名数组

var c = node.className; // 节点的类名

var i; // 循环计数器

// 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。

When a function is to be invoked immediately, the entire invocation expression should be wrapped in parens so that it is clear that the value being produced is the result of the function and not the function itself.

var collection = (function () {

var keys = [], values = [];

return {

get: function (key) {

var at = keys.indexOf(key);

if (at >= 0) {

return value[at];

}

},

set: function (key, value) {

var at = keys.indexOf(key);

if (at < 0) {

at = keys.length;

}

keys[at] = key;

value[at] = value;

},

remove: function (key) {

var at = keys.indexOf(key);

if (at >= 0) {

keys.splice(at, 1);

value.splice(at, 1);

}

}

};

}());

**7、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](http://javascript.crockford.com/private.html)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](http://yuiblog.com/blog/2006/11/13/javascript-we-hardly-new-ya/)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)

**8、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](http://yuiblog.com/blog/2006/04/11/with-statement-considered-harmful/)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。

**9、其它建议**

**{} 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。