Javascript 编码规范

hellohtml5.com

版本记录

Javascript 编码规范 1.0	1.0	hellohtml5	2011.5.7

前言

这是一组 Javascript 编码规范 ,借鉴了 Sun 公司的官方文档《Java 编码规范》。

一个软件的生命期与其代码质量成绝对正比。软件在生存期间，可能要被上百的人修改或查看。如果开发者很容易了解到软件的架构和特性，那么在近期对其进行修改时出现问题的几率就比较低。

代码规范可以增强代码的健壮性。

Javascript 代码是公开的 ,因此代码应具备发布的质量。

代码可以更整洁。

文件规范

Javascript 代码应以单独.js 文件的形式存储和输出。

Javascript 代码不应该直接嵌入 HTML 文件，除非此代码在一次性的会话中用到。在 HTML 中使用 JS 代码会增加页面的体积，并无法使用缓存和压缩机制。

<script src=filename.js>标签应放在<body>标签的尽可能底部。这样做可以尽可能快的使页面显示出来。没有必要使用 language 或者 type 属性。

缩进规范

缩进的单位为 4 个空格。避免使用 tab，因为 tab 并没有统一的标准规定其宽度。使用空格会增加文件大小，但是后期可以通过压缩 JS 代码的方式除去此影响。

行长度规范

避免单行代码超过 80 个字符。如果语句不能在一行内完成，那么应该打断它。在操作符后打断，一般在分号后打断。在操作符后打断可以降低复制粘贴代码的操作导致出错的可能性。下一行应该以 8 个空格开始。

注释规范

尽量为你的代码进行注释，这是非常有用的，当有人(可能是你自己)需要知道你都做了些什么。注释应该良好而整洁的编写，就想是对代码的批注。稍微带点儿幽默最好了，抱怨和沮丧就算了。

非常重要的一点是注释应该及时更新。错误的注释可能使程序更加难以理解。

注释应言之有物，主要说明不能立即从代码中发现的东西。不要浪费你和别人的时间在如下的注释上。

```
i = 0;  // 把 i 赋值为 0
```

一般使用行注释。块注释用来做正规的文档和块代码注释。

变量声明规范

所有的变量应该在使用前声明。Javascript 语言并不要求这样做，但是这样做可以使代码易于阅读，并且易于检查出未被声明的变量或许这些变量已经是隐含全局变量。永远不要使用隐含全局变量。

var 语句应该永远是函数体的第一个语句

每个变量声明语句在最好在单独的行并进行注释。这些变量应以字母表的顺序进行排序。

```
var currentEntry; // 当前表格入口
var level;        // 缩进层级
var size;         // 表格大小
```

尽量避免使用显式的全局变量，绝对不要使用隐含全局变量。

函数声明规范

所有的函数应该在使用前声明。内部函数(函数内部的函数)应该用 var 关键字声明。这样对明确那些变量在此函数作用域有帮助。

函数名和左括号之间不应该有空格。右括号和{之间应有一空格。函数体缩进 4 个空格。}与函数定义起始处对齐。

```
function outer(c, d) {
    var e = c * d;
    function inner(a, b) {
        return (e * a) +b;
    }
    return inner(0, 1);
}
```

这样做在代码结构非常复杂时也可以提供良好的可读性。

```
function getElementsByClassName(className) {
    var results = [];
    walkTheDOM(document.body, function (node) {
        var a ;                          // classname 数组

        var c= node.className; // node 的 classname

        var i;                           // 计数

// 如果 node 有 className 属性 ,那么分割为较小的 classname

// 如果有匹配到指定的 classname 那么保存到结果集合里

        if (c) {
            a = c.split(' ');
            for (i = 0; i < a.length; i+=1) {
                if (a[i] === className) {
                    results.push(node);
                    break;
                }
            }
        }
    });
    return results;
}
```

如果是匿名函数，在 function 和(之间应有一空格。如果忽略这个空格，那么看起来这个函数的名字就是 function，这样会造成误读。

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

尽量不要使用全局函数。

立即执行的匿名函数应该用一对括号包裹起来，清楚的表明我们要使用的是函数执行的结果，而不是函数自身。

```
var collection = (function () {
    var keys = [], values = [];

    return {
        get: function (key) {
            var at = keys.indexOf(key);
            if (at >= 0) {
                return values[at];
            }
        },
        set: function (key, value) {
            var at = keys.indexOf(key);
            if (at < 0) {
                at = keys.length;
            }
            keys[at] = key;
            values[at] = value;
        },
        remove: function (key) {
            var at = keys.indexOf(key);
            if (at >= 0) {
                keys.splice(at, 1);
                values.splice(at, 1);
            }
        }
    };
}());
```

命名规范

变量名应以 A-Z，a-z，0-9，和_组成，不要在变量名里使用$和\等特殊符号。

尽量不要使用_作为变量名的起始字符，用_开始的变量暗指此变量为私有变量。

绝大多数的变量应以小写字母开头。

当一个函数只能作为构造函数被调用时(使用 new 关键字)，变量名应以大写字母开头。当看到一个函数以大写字母开头时，我们就知道这个函数应该以构造函数的形式调用。

全局变量全部大写。

语句规范

简单语句

简单语句独占一行，并以;结尾。注意，使用赋值语句给变量赋值一个函数或者实例仍然是一个赋值语句，必须以;结尾。

复合语句

复合语句是指包含在{}里的语句

1. 复合语句应缩进 4 个空格。
2. {应在开始复合语句的行尾。
3. }应独占一行，并与起始复合语句的那行头部对齐
4. {}应包裹所有复合语句，即使这些复合语句只有一行。比如 if 或 for 语句，这样可以方便的添加新语句，而不意外引入 bug。

return 语句

使用 return 语句返回值，不要用( )包裹该值。

if 语句

if 语句采用如下的样式声明。

```
if (condition) {
    statements
}

if (condition) {
    statements
} else {
    statements
}

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

for 语句

for 语句采用如下的样式声明。

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

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

while 语句

while 语句采用如下的样式声明。

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

do 语句

do 语句采用如下的样式声明。

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

do-while 语句应以;结尾

switch 语句

switch 语句采用如下的样式声明。

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

case 语句应与 switch 语句对齐。每个 case 语句流程必须以 break, return, throw 之一结尾。default 除外。

try-catch 语句

try-catch 语句采用如下的样式声明。

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

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

continue 语句

尽量不要使用 continue 语句，continue 会使整个控制流程变得晦涩。

with 语句

不要使用 with 语句。

空白使用规范

对逻辑相关的代码块使用空白行隔开可以增强代码的可读性。

使用空白字符的一些注意事项：

1. 语法关键字与(应该用一个空格隔开。

```
while (true) {
```

2. 空白字符不应该在函数名字和(之间使用，以区分函数定义和函数调用。
3. 所有除.（ [之外的二元操作符应在其操作数之间插入一空白。
4. 所有一元操作符与其操作数之间不应插入空白除非该一元操作符是语法词，如 typeof instanceof。
5. for 语句中的循环控制部分中的;后应留一空白。
6. ,后总是跟一个空白。

其它规范

{}和[]

总是用{}代替 new Object() ,总是用[]代替 new Array()。

当成员名为有序的数字时使用数组 Array，当成员名为任意字符时使用 Object 来表达数据。

,操作符

避免使用,操作符。以下几种情况除外：

1. for 语句循环控制流程语句。
2. object 声明的语法。
3. array 声明的语法。
4. var 声明变量的语法。
5. 参数列表。

赋值表达式

避免在条件判断中进行赋值操作。

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

还是

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

是正确的？避免使用那些不是很容易被人认为是“正确的”的代码。（尽量采用地球人的方式进行条件判断）

===和!==操作符

大部分情况下，使用===和!==是更好的方法。==和!=进行了数据类型强制转换。具体来讲，不要使用==来跟布尔值判断。

++操作符

注意不要在+之后再跟一个+或者++。这种方式会使人迷惑，用()包裹他们会使你的真实目的更加清楚。

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

写成

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

更好些。这样+ +就不会被误读为++。

eval

eval 函数是最多被滥用的 Javascript 特性。避免使用它。

Function

不要使用 Function 构造函数(即 new Function)。

setTimeout 和 setInterval

不要在 setTimeout 或 setInterval 函数中传入字符串。

参考资料

[1]. Code Conventions for the JavaScript Programming Language