中国移动通信
CHINA MOBILE

人工智能与智慧运营研发中心
软件开发规范

JavaScript 编码规范

版本号：1.1

2019-10-10

人工智能与智慧运营研发中心

目 录

一、 代码总体原则

1. 清晰第一

清晰性是易于维护、易于重构的程序必需具备的特征。代码首先是给人读的，其次才给机器用来执行。

目前软件维护期成本占整个生命周期成本的 40%~90%。根据业界经验，维护期变更代码的成本，小型系统是开发期的 5 倍，大型系统（100 万行代码以上）可以达到 100 倍。业界的调查指出，开发组平均大约一半的人力用于弥补过去的错误，而不是添加新的功能来帮助公司提高竞争力。可见代码清晰的重要。

2. 简洁为美

简洁就是易于理解并且易于实现。代码越长越难以看懂，也就越容易在修改时引入错误。写的代码越多，意味着出错的地方越多，也就意味着代码的可靠性越低。因此，我们提倡大家通过编写简洁明了的代码来提升代码可靠性。废弃的代码(没有被调用的函数和全局变量)要及时清除，重复代码应该尽可能提炼成函数。

3. 风格一致

产品所有人共同分享同一种风格所带来的好处，远远超出为了统一而付出的代价。在公司编码规范的指导下，审慎地编排代码以使代码尽可能清晰，是一项非常重要的技能。新的源文件使用新的规则编码代码，老的源文件代码修改可以继续沿用老规则。

二、 范围

本规范制定了编写 JAVASCRIPT 语言程序的基本原则、规则和建议。

本规范适用于公司内使用 JAVASCRIPT 语言编码的所有软件/项目。本规范自发布之日起生效，对以后新编写的代码应遵守本规范。老的源文件代码修改可以继续沿用老规则。

本规范实施中遇到问题，可以反馈给海报中说明的人员。

在某些情况下需要违反本文档给出的规则时，相关团队必须通过一个正式的流程来评审、决策规则违反的部分。

三、 命名

1. 变量和函数

[1] 变量名由 26 个大小写字母(A..Z,a..z),10 个数字(0..9),和_(下划线)组成。

[2] 变量和函数名使用驼峰式（PascalCase）。

[3] 避免无意义命名（asdf）。

[4] 避免抽象命名（getData）。

[5] 避免单字母命名（a）。

[6] 避免拼写错误变量名（sesionStorage）。

[7] 避免自定义缩写（CFF-CovertFileFormat）。

[8] 避免混用英文和拼音。

[9] 避免使用国际化字符(如中文)。

[10]避免使用\(反斜杠)。

[11]_和$符号可以用作特殊用途，不做控制，但是避免_, __, ___或$, $$, $$$这类无意义变量名。

[12]重要的私有变量请使用私有成员的形式。

[13]变量名和方法名首字母小写。

[14]与 new 配合使用的构造函数名首字母大写。

[15]全局变量全部大写。

2. 文件

[1] 同一项目文件命名规则保持统一。

例如：

[kebab-case 烤串式文件名][文件类型(可选)][文件扩展名]

my-app-tool.service.js

primary-button.css

[2] JavaScript 程序独立保存在后缀名为.js 的文件中。

[3] 文件编码请使用 UTF-8。

[4] 线上需要将 js 压缩成 min.js。

四、 代码格式

1. 缩进

[1]采用 4 个空格缩进，禁止使用 tab 字符。

[2] 出现长的方法链（>2 个）时用缩进。用点开头强调该行是一个方法调用，而不是一个新的语句。

例如：

```
// bad
$('#items').find('.selected').highlight().end().find('.open').updateCount();
// good
$('#items')
    .find('.selected')
    .highlight()
    .end()
    .find('.open')
    .updateCount();
```

2. 空格和空白行

[1] 用空格隔开运算符。例如：

```
// bad
const x=y+5;
// good
const x = y + 5;
```

[2] 一元操作符与其操作数之间不要有空格,除非操作符是个单词,比如 typeof。

[3] 语句控制部分,比如 for 语句中的; (分号)后须加一个空格。

[4] 圆括号()、方括号[]里前后不要加空格，花括号{}里前后加空格。例如：

```
// bad
function bar( foo ) {
    return foo;
}
// good
function bar(foo) {
    return foo;
}

// bad
const foo = [ 1, 2, 3 ];
console.log(foo[ 0 ]);
```

```
// good，  逗号分隔符还是要空格的
const foo = [1, 2, 3];
console.log(foo[0]);

// bad
const foo = {clark: 'kent'};
// good
const foo = { clark: 'kent' };
```

[5] 作为语句的花括号内也要加空格 —— “{“ 后和 “}”前都需要空格。例如：

```
// bad
function foo() {return true;}
if (foo) {bar = 0;}
// good
function foo() { return true; }
if (foo) { bar = 0; }
```

[6] 逗号前不加空格，逗号后需加空格。例如：

```
// bad
var foo = 1,bar = 2;
var arr = [1 , 2];
// good
var foo = 1, bar = 2;
var arr = [1, 2];
```

[7] 调用函数时，函数名和小括号之间不要空格。例如：

```
// bad
func ();
// good
func();
```

[8] 在对象的字面量属性中， key 和 value 之间要有空格。例如：

```
// bad
var obj = { “foo” : 42 };
var obj2 = { “foo”:42 };
// good
var obj = { “foo”: 42 };
```

[9] 行末不要空格。

[10]逻辑块间空一行，不要空多行。例如：

```
// bad
if (foo) {
    return bar;
}



return baz;
```

```
// good
if (foo) {
    return bar;
}

return baz;
```

[11]不要用空白行填充块。例如：

```
// bad
function bar() {

    console.log(foo);

}
```

```
// good
function bar() {
    console.log(foo);
}
```

3. 每行长度

单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：

[1] 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。

[2] 运算符与下文一起换行。

[3] 方法调用的点符号与下文一起换行。

[4] 方法调用中的多个参数需要换行时，在逗号后进行。

[5] 在括号前不要换行，见反例。

例如：

```
// 超过 120 个字符的情况下，不要在括号前换行
sb.append(“Jack”).append(“Ma”)...append
    (“alibaba”);
// 参数很多的方法调用可能超过 120 个字符，不要在逗号前换行
method(args1, args2, args3, ...
     , argsX);
//good
StringBuilder sb = new StringBuilder();
// 超过 120 个字符的情况下，换行缩进 4 个空格，点号和方法名称一起换行
sb.append(“Jack”).append(“Ma”)...
    .append(“alibaba”)...
    .append(“alibaba”)...
    .append(“alibaba”);
```

4. 变量声明

[1] 所有变量在使用前必须先声明。

[2] 所有变量定义放在函数首部。

[3] 每个变量声明单独一行，注释说明，按字母排序。

例如：

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

[4] 避免有声明但未使用的变量

[5] 全局或整个函数体都要用到的变量全部前置声明，查看清晰。

[6] 局部或者临时变量就近声明，减少干扰。

[7] 避免定义过多的全局变量，变量过多时建议将相关的变量放到一个对象上。

[8] 避免局部变量覆盖全局变量。

5. 函数声明

[1] 所有函数在调用前先声明。

[2] 内函数声明紧跟变量声明之后。

[3] 函数名与左括号“(“之间不要空格。

[4] 右括号”)”与函数主体的左大括号”{“之间插入一个空格。

[5] 函数体缩进 4 个空格。

[6] 右大括号”}”与函数声明首行对齐。

例如：

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

[7] 推荐以下函数声明方式，保证内联函数与混合结构可读性最好：

```
function getElementsByClassName(className) {
    var results = [];
    walkTheDOM(document.body, function (node) {
        var a;                    //
        var c = node.className; //
        var i;                    //
        if (c) {
            a = c.split(' ');
            for (i = 0; i < a.length; i += 1) {
                if (a[i] === className) {
                    results.push(node);
                    break;
                }
            }
        }
    });
    return results;
```

```
        }
```

[8] 匿名函数关键字 function 和左括号”(“之间要插入一个空格。

例如：

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

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

[9] 尽量不使用全局函数。

五、 控制语句

1. 简单语句

[1] 每行只含一条语句。

[2] 语句结尾加分号(;)。

2. 复合语句

复合语句是包含在大括号{}中的语句。

[1] 左大括号”{”应在复合语句实行结尾处。

[2] 被大括号括起内容多缩进四个空格。

[3] 右大括号”}”应与左大括号所在行首对齐。

[4] 控制语句（例如 if 语句/for 语句）必须用大括号，即使内容只有一条语句。

3. 标示

while, do,for,switch 语句必须被标示。

4. 具体语句参考格式

[1] return 语句

a) return 语句中不要使用括号括住返回值

b) 返回表达式时，表达式应与 return 关键字在同一行。

[2] if 语句参考格式

```
if (condition) {
    statements;
}

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

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

[3] for 语句参考格式

a) 用于已经知道相关参数的数组循环。

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

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

b) 应用于对象，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。

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

```
        }
```

[4] while 语句参考格式

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

[5] do 语句参考格式

注意：do 语句要以分号”;”结尾

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

[6] switch 语句参考格式

a) case 与 switch 对齐。

b) 除 default 外，每组 statements 以 break、return、throw 结尾。

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

[7] try 语句参考格式

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

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

[8] continue、eval 尽量少用，使用时需注释必要性。

[9] with 不要用。

六、 注释

[1] 描述是什么及为什么。

[2] 避免注释冗长。

[3] 及时更新注释。

[4] 重点标注在复杂难懂的逻辑上。

[5] 注释正式文档和无用代码选择块注释。

[6] 具体可参考【PHP 编码规范】注释部分。

七、 其他

1. 数组

[1] 创建对象数组时，尽量使用字面量。

例如：

```
// good
var obj = {};
var arr = [];
// bad
var obj = new Object();
var arr = new Array();
```

[2] 若成员为有序数字用数组保存，无序字符串用对象保存。

[3] 若数组有多行，在数组的 [后和] 前断行。

2. 操作符

[1] 除 for 语句控制部分等特定情况，避免使用逗号”,”操作符。

[2] 用 === 和 !== ，不用 == 和 !=

[3] 在”+”后跟”+”或”++”时，用括号隔开。

例如：

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

3. 作用域

在 JavaScript 中块没有域。只有函数有域。除复合语句外不要使用块。

4. 赋值表达式

避免在 if 和 while 语句的条件部分进行赋值。

```
//bad
if (a = b) {
//good
if (a == b) {
```

修订记录

版本	人员	修订情况
V1.0	金镝　许堃杰　徐惠群	2019-09-25 初稿
V1.1	徐惠群	2019-10-10 格式修订