Javascript 代码规范

目录

文件命名及其内部结构

文件/资源命名

以可读性而言，减号（-）是用来分隔文件名的不二之选。
请确保文件命名总是以字母开头而不是数字。而以特殊字符开头命名的文件，一般都有特殊的含义与用处（比如 compass[1] 中的下划线就是用来标记跳过直接编译的文件用的）。

资源的字母名称必须全为小写，这是因为在某些对大小写字母敏感的操作系统中，当文件通过工具压缩混淆后，或者人为修改过后，大小写不同而导致引用文件不同的错误，很难被发现。

还有一些情况下，需要对文件增加前后缀或特定的扩展名（比如 .min.js, .min.css），抑或一串前缀（比如 3fa89b.main.min.css）。这种情况下，建议使用点分隔符来区分这些在文件名中带有清晰意义的元数据。

不推荐

```
1. MyScript.js
2. myCamelCaseName.css
3. i_love_underscores.html
4. 1001-scripts.js
5. my-file-min.css
```

推荐

```
1. my-script.js
2. my-camel-case-name.css
3. i-love-underscores.html
4. thousand-and-one-scripts.js
5. my-file.min.css
```

Javascript 文件

JavaScript 程序应该作为一个.js 文件存储和发布。
JavaScript 文件应该尽量放置到统一的 js 文件夹下，如有特殊需要可做相应调整。
JavaScript 代码不应该嵌入在 HTML 文件里，除非那些代码是一个单独的会话特有的。<script src=filename.js>标签应该在 body 里越靠后的位置越好。这减少了由于加载 script 而导致的其它页面组件的延迟。没有必要使用 language 或者 type 属性。由服务器而不是 script 标签来决定 MIME 类型。

缩进

缩进的最小单位是 2 个空格。不支持用 TAB 键进行缩进。这是因为直到现在还没有统一的标准来定义 TAB 键所代替的空白大小，有些编辑器解析为 4 个空格大小，有些则解析为 8 个。因而用不同的编辑器查看代码，可能造成格式混乱。使用空格会引起文件变大，但是这点大小对局域网无关紧要，而且差别被 minification 消除了。

换行

每个独立语句结束后必须换行。不要让一行代码超过 120 个字符。当一条语句不能在单独一行写完时，可能有必要拆分它。运算符处换行时，运算符必须在新行的行首。不允许在 ， 或 ； 前换行。下一行应该增加一个缩进。

空格

空行通过将逻辑相关的代码放到一起来增加可读性。
空格应该用于如下情况：
1，关键字后面跟“(”(左圆括号)时应该用一个空格隔开。

```
1. while (true) {
```

2，方法名和方法的“(”(左圆括号)之间不要有空格。这利于区分关键字和方法调用。
3，所有的二元操作符，除了“.”(圆点)、“(”(左圆括号)和“[”(左中括号)，都应该使用一个空格来和操作数隔开。
4，一元操作符和操作数之间不应该使用空格隔开，除了操作符是一个单词时，如 typeof。
5，for 语句控制部分的每个“;”(分号)应该在后面跟一个空格。
6，每个“,”(逗号)后面应该跟一个空格。

注释

慷慨的写注释。留下一些供需要理解你做了什么的人们(可能是你自己)下次阅读的信息是有用的。注释应该书写良好和清晰，就像它们 标注的代码一样。偶尔小幽默一把也是可以的。挫折和怨恨就别写了。
更新注释非常重要。错误的注释让程序更难读懂和理解。
让注释有意义。更多的关注于不能马上可见的东西。不要用如下内容浪费读者的时间：

```
1. i = 0; // 置 i 为 0
```

一般使用行注释。把块注释用于正式文档或外部注释。
行注释为“//”，块注释为“/* .... */”，块注释用作对整个代码段的注销，或较正式的声明中，如函数参数、功能、文件功能等的描述中。

声明及命名

变量声明

所有的变量应该在使用前声明。
var 语句应该为方法体内的第一个语句。每个变量声明应该自己占一行并有注释。它们应该按字母顺序排列。
建议只用一个 var 关键字声明，多个变量用逗号隔开，并把赋值尽量写在变量申明中。如：

```
1. var a = 10,
2.     b = 10,
3.     c = 100;
```

方法声明

所有的方法应该在它们使用前声明。内部方法应该位于 var 语句后面。
方法名和参数列表的“(”(左圆括号)之间不应该有空格。在“)”(右圆括号)和“{”(左大括号)之间有一个空格。
方法体本身缩进 2 个空格。“}”(右大括号)应该和方法声明处对齐。如：

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

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

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

如果一个方法字面量为匿名的，则在“function”和“(”(左圆括号)之间应该有一个空格。如果省略空格，则它可能看起来方法名是“function”，而这是错误的。

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

尽量少用全局方法。

命名

命名应该由 26 个大小写字母(A .. Z, a .. z)，10 个数字(0 .. 9)和_(下划线)组成。
命名应该采用有意义的单词，否则难于维护，如果需要压缩版本的 js，可以使用 js 压缩工具进行压缩生成 min.js。
大多数变量和方法名应该以小写字母开始。
必须使用 new 前缀的构造函数应该以大写字母开始。JavaScript 不会在省略 new 时报编译期警告或运行时警告。
不使用 new 时会发生坏事情，所以大写首字母规范是我们拥有的唯一的防御。
全局变量应该全部使用大写字母。(JavaScript 没有宏或常量，所以没有多少要求使用大写字母来表示 JavaScript 的特性的场景)

Javascript 基础语句以及特殊语句

基础语句

简单语句

每行应该包含至少一个语句。在每个简单语句末尾添加一个“;”(分号)。

复合语句

复合语句是包含一个用“{}”(大括号)包围语句列表的的语句。
1，包围的语句应该再缩进 2 个空格。

2，“{”(左大括号)应该位于开始复合语句的行的末尾。
3，“}”(右大括号)应该新起一行并且和相匹配的“{”所在那行的起始位置对齐
4，当语句是控制结构的一部分时，所有语句都应该用括号包围，即使是单行语句，例如 if 或 for 语句。这让添加语句更容易而且不会引起 bug。

标签

语句标签是可选的。只有如下语句需要被标签标识：while，do，for，switch。

return 语句

具有值的 return 语句不应该使用“()”(圆括号)包围值。返回值表达式必须和 return 关键字在同一行从而避免插入分号。

if 语句

if 语句应该使用如下格式：

```
if (condition) {
  statements;
}

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

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

for 语句

for 语句应该使用如下格式：

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

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

第一种格式应该和数组使用。

第二种格式应该和对象使用。注意添加到对象的 prototype 中的成员将被包含在遍历中。通过使用 hasOwnProperty 方法来区分对象的
成员是明智的:

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

while 语句

while 语句应该使用如下格式:

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

do 语句

do 语句应该使用如下格式:

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

不像其它复合语句，do 语句始终使用“;”(分号)结尾。

switch 语句

switch 语句应该有如下格式:

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

每组语句(除了 default)应该以 break，return 或者 throw 结束。不要 fall through。
但是有些地方建议：不要使用 switch，switch 在所有的编程语言中都是个非常错误的难以控制的语句，建议用 if else 来替换它。

try 语句

try 语句应该使用如下格式:

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

```
6.
7. try {
8.   statements;
9. } catch (variable) {
10.   statements;
11. } finally {
12.   statements;
13. }
```

continue 语句

不要使用 continue 语句。它会让方法的控制流程模糊。

with 语句

不要使用 with 语句。

特性检测而非浏览器检测

一些代码是写来发现浏览器版本并基于用户正使用的客户端的对其执行不同行为。这个，总的来说，是一个非常糟的实践。更好的方法是使用特性检测，在使 用一个老浏览器可能不支持的高级的特性之前，首先检测（浏览器的）是否有这个功能或特性，然后使用它。
例子：

```
1. if (document.getElementById) {
2.    var element = document.getElementById('MyId');
3. } else {
4.    alert('Your browser lacks the capabilities required to
    run this script!');
5. }
```

避免 document.all

document.all 是由 Microsoft 的 IE 所引进的，并不是一个标准的 Javascript DOM 特性。

避免同步的 ‘ajax’ 调用

使用 JSON

当需要将数据结构存储成纯文本，或者通过 Ajax 发送/取回数据结构，尽可能使用 JSON 代替 XML。

额外建议事项（非必须遵守）

额外的建议

字符串

统一使用单引号(‘)，不使用双引号(“)。这在创建 HTML 字符串非常有好处：

```
1.     var msg = 'This is some HTML <div class="makes-sense"></div>';
```

{}和[]

使用{}替代 new Object()。使用[]替代 new Array()。
当成员名字为连续的整数时使用数组。当成员名字为任意的字符串或名字时使用对象。

逗号操作符

不要使用逗号操作符，除了 for 语句的控制部分的严格使用。(这不适合逗号操作符，它应该用于对象字面量，数组字面量，var 语句和参数列表。)

块作用域

在 JavaScript 里块没有作用域，只有方法有作用域。不要使用块，除了复合语句一定需要用到外。

赋值表达式

不要在 if 和 while 语句的条件部分做赋值。不要写不易懂的代码。

===和!==操作符

始终使用===和!==操作符会更好。==和!=操作符会做类型强制转换。因而，'=='进行的判断结果可能产生偏差。'!='与'!=='的区别亦是如此。本文提倡尽量使用'==='来进行逻辑等的判断，用'!=='进行逻辑不等的判断。特别是，不要使用==来和“假”值做比较。

令人混淆的加和减

注意不要在“+”后面跟“+”或“++”。这种模式令人混淆。在它们之间插入圆括号来让你的意图更清晰。

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

// is better written as

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

这样“+ +”就不会被读错成“++”。

邪恶的 eval

eval 方法是 JavaScript 里最滥用的特性。不要使用它。
eval 有别名。不要使用 Function 构造函数。不要传递字符串给 setTimeout 或者 setInterval。