JavaScript编程规范

编写		日期	2023年02月01日
审核		日期	
批准		日期	

北京□□□□□□□□□□□□□□□□□□

（内部资料，注意保密）

修改记录

版本	日期	修改纪要	修改人
V0.1	2023-02-01	创建	

目 录

1. 概述

1.1 简介

人民邮电出版社在2013年04月出版了Nicholas C. Zakas编写的《编写可维护的JavaScript》第一版，该书主要参考了Java语言编程规范、Google JavaScript编程规范、Dojo编程风格指南、Dauglas Crockford的JavaScript编码规范、jQuery核心风格指南等文档。本文档主要参考该文档，针对该书提出的一些A可以B也可以等存在多个可选项的情况下，提出北京□方□数软件二部自己的JavaScript编程规范，**所有相关开发人员都必须遵照该规范来执行检查**。

编程规范包含编程风格（style guideline）和编程实践两部分。编程风格主要关注的是代码的呈现，编程实践主要关注的是编码的结果，也可以看做是秘方。

1.2 适用范围

当编写Web前端（利用jQuery、React.js等）或后端（利用Node.js等）编写JavaScript相关代码时，需要遵照该规范来执行检查。

1.3 缩写和术语

略。

1.4 参考资料

[1] Nicholas C. Zakas, 编写可维护的JavaScript, 人民邮电出版社, 2013年04月.

[2] 书写具备一致风格、通俗易懂 JavaScript 的原则, https://github.com/rwaldron/idiomatic.js/tree/master/translations/zh_CN

2. 编程风格之基本格式化

如果已经存在一个通用编码规范，它必须受到尊崇。

"对风格的挑刺毫无意义可言。它们必须是指导原则，且你必须遵循。"

Rebecca Murphey

"成为一个优秀的成功项目管理者的一个条件是，明白按自己的偏好风格写代码是非常不好的做法。如果成千上万的人都在使用你的代码，那么请尽可能通俗易懂地写出你的代码，而非在规范之下自作聪明地使用自己偏好的风格。"

Idan Gazit

编程风格指南的核心是基本的格式化规则（formatting rule）。

2.1 空白

永远都不要混用空格和Tab，只允许使用空格，不能使用TAB。

开始一个项目，在写代码之前，选择软缩进（空格），并将其作为最高准则。

在编辑器（NotePad++、Sublime、Netbeans、phpStorm等）中，请总是打开“显示不可见字符”这个设置。好处是：

- 保证一致性
- 去掉行末的空格
- 去掉空行的空格
- 提交和对比更具可读性

2.2 缩进（zakas 1.1与A.1）

每一行的层级由4个空格组成，不能使用TAB进行缩进。

2.3 大括号的位置（zakas 1.2）

起首的大括号跟在关键字的后面，避免分析器的自动分号插入（Automatic Semicolon Insertion, ASI）机制产生错误。

举例（错误写法）：

```
block
{
    ...
}
```

举例（正确写法）：

```
block {
    ...
}
```

2.4 行的长度（zakas 1.3,1.4与A.2）

单行长度不应当超过 80个字符。如果一行多于80个字符，应当在一个运算符（逗号，加号等）后换行，下一行增加2个层级缩进（8个字符）。

该规则的一个例外：当给变量赋值时，第二行的位置应当和赋值运算符的位置保持对齐。

举例：

```
var result = something + anotherThing + yetAnotherThing + somethingElse +
                anotherSomethingElse
```

2.5 空格（zakas A.4与A.5）

二元运算符（赋值运算符和逻辑运算符）前后必须使用一个空格来保持表达式的整洁。

当使用括号时，紧接左括号之后和紧接右括号之前不应当有空格。

2.6 空行（zakas 1.5）

空行是提高代码可读性的强大武器。推荐在以下场景中添加空行：

- 在方法之间
- 在方法中的局部变量和第一条语句之间。
- 在多行或单行注释之前。
- 在方法内的逻辑片段之间插入空行，提高可读性。

2.7 基本类型-字符串（zakas 1.7.1）

在PHP中，字符串尽可能使用单引号替代双引号（双引号内的变量需要解析），以避免PHP搜索字符串中的变量导致的性能下降。而在javaScript中，单引号与双引号功能没有区别。为了使两种语言相互切换保持一致，建议应当始终使用单引号（避免使用双引号）且保持一行，且新增js片段代码从头到尾只保持一种风格。

避免在字符串中使用斜线另起一行，可用字符串连接符（+）将字符串分成多行。

举例（不好的写法）：

```
var lingString = “Here is a story, of a man \
                named Brady.”;
```

举例（推荐的写法）：

```
var lingString = “Here is a story, of a man ” +
                “named Brady.”;
```

2.8 基本类型-数字（zakas 1.7.2）

不要省略小数点之前或之后的数字。

禁止使用八进制数字写法。

2.9 基本类型-null（zakas 1.7.3）

null 的最佳解释是当做对象的占位符（placeholder）。可以在以下场景中使用特殊值null：

- 用来初始化一个变量，该变量可能被赋值为对象。

- 用来和一个已经初始化的变量进行比较，这个变量可以是非对象也可以是对象。
- 当函数的参数期望是对象时，用作参数传入。
- 当函数的返回值期望是对象时，用作返回值传出。

以下场景不应当使用 null：

- 不要使用null来检测是否传入了某个参数。
- 不要用null来检测一个未初始化的变量。

2.10 基本类型-undefined（zakas 1.7.4）

避免在代码中使用特殊值undefined，判断一个变量是否定义应当使用typeof操作符。

2.11 对象字面量（zakas 1.7.5与A.6）

使用对象字面量（object literal syntax）来创建对象，应当使用如下格式：

- 起始左花括号应当同表达式保持同一行。
- 每个属性（property）的key:value应当保持一个缩进，第一个属性应当在左花括号后另起一行。
- 每个属性（property）的key:value应当使用不含引号的key, 其后紧跟一个冒号（之前不含空格），而后是一个空格，接下来是value。
- 如果属性值是函数类型，函数体应当在属性名（或key）之下另起一行，而且函数体前后均应保留一个空行。
- 一组相关的属性前后可以插入空行以提升代码的可读性。
- 结束的右花括号应当独占一行。

当对象字面量作为函数参数时，如果值是变量，起始花括号应当同函数名在同一行，其余先前列出的规则同样适用。

2.12 数组字面量（zakas 1.7.6）

不要显式地适用Array构造函数来创建数组。

推荐使用两个方括号将数组初始元素括起来，数组项之间用逗号隔开。

3. 编程风格之注释

对于代码的总体维护性而言，注释是非常重要的一环，它是另一种形式的文档。

3.1 哪些情况下应当使用注释（zakas 2.3与2.4）

以下这些情况应当使用注释：

- 代码晦涩难懂。

- 可能被误认为错误的代码。
- 必要但并不明显的针对特定浏览器的代码。
- 对于对象、方法或者属性，生成文档时有必要的。

3.2 单行注释（zakas 2.1与A.7.1）

单行注释以两个斜线开始，以行尾结束。**双斜线后要有一个空格**，用来让注释文本有一定的偏移。

单行注释有三种使用方式：

- 独占一行，用来解释下一行代码。并且**这行注释之前总有一行空行，缩进的层级与下一行代码保持一致**。
- 在代码行的尾部注释，用来解释它之前的代码。**代码与注释之间至少有一个缩进。注释包括之前本行的代码不应该超过单行最大字符数80限制。**
- 注释掉大块代码。

3.3 多行注释（zakas 2.2与A.7.2）

多行注释以/*开始，以*/结束。

推荐使用 Java 风格的多行注释，每个多行注释至少包含三行：

第一行仅仅包含/*，与描述代码保持相同层次的缩进，不应当有其他文字。

第二行是以*开始且和上一行的*保持左对齐，这些行可以有文字描述。

最后一行是*/切和先前行保持对齐，也不应当有其他文字。这种注释格式如下：

```
/*
 * 这里是注释
 * 这里是注释
 * 这里是注释
 */
```

每一个多行注释之前应当预留一个空行。

代码尾部注释不要使用多行注释格式。

3.4 注释声明（zakas A.7.3）

注释有时候用于给一段代码声明额外的信息。这些声明的格式以单个单词打头并紧跟一个冒号。可使用的声明如下：

TODO: 说明代码还未完成，应当包含下一步要做的事情。

HACK: 表明代码实现走了一个捷径，应当描述使用hack的原因。

XXX：代码存在问题并应当尽快修复。

FIXME：代码存在问题并应当尽快修复，重要性略次于XXX。

REVIEW: 说明代码任何可能的改动都需要评审。

这些声明可能在单行注释或多行注释中使用，并且应当遵循与注释类型相同的格式规则。

4. 编程风格之变量、函数和运算符

4.1 变量的命名（zakas 1.6.1与A.7.6）

变量命名应当采用小驼峰式（Camel Case）命名格式：首字母小写，后续每个单词首字母大写。

变量名称的第一个单词应当是一个名词。

尽量在变量名中体现出值的数据类型：例如count、length、size表明数据类型是数字，而name、title、message表明数据类型是字符串。

尽量在变量名中体现出值的单位：例如磁盘空间大小单位bit、byte、Mb、Gb等，时间单位sec、ms等。

变量名称中最好不要使用下划线。

4.2 函数的命名（zakas 1.6.1与A.7.6）

函数命名应当采用小驼峰式（Camel Case）命名格式：首字母小写，后续每个单词首字母大写。

函数名称的第一个单词应当是动词。

常见动词约定：

can：函数返回一个boolean值

has：函数返回一个boolean值

is：函数返回一个boolean值

get：函数获取一个非boolean值

set：函数用来保存一个值

函数名称中最好不要使用下划线。

4.3 常量的命名（zakas 1.6.2与A.7.6）

常量（值不会被改变的变量）的命名应当是所有字母大写，不同单词之间用单个下划线隔开。

构造函数的命名采用大驼峰式命名格式：首字母大写。

4.4 构造函数的命名（zakas 1.6.3与A.7.6）

JavaScript中构造函数就是通过new运算符创建对象的函数。

构造函数的命名采用大驼峰式命名格式：首字母大写。

构造函数的名称的第一个单词常常是名词，因为是用来创建某个类型的实例的。

举例（推荐的写法）：

```
function Person(name) {
   this.name = name;
}

Person.prototype.sayName = function() {
   alert(this.name);
}

var me = new Person(“”Nicholas”);
```

4.5 变量声明（zakas 4.1与A.7.4）

所有的变量在使用前都应当先定义。

变量定义应当放在函数开头，使用单个var表达式，这样可使代码更短，下载更快。

每行一个变量，每个变量之间以逗号分隔。

除了首行，所有航都应当多一层缩进以使变量名能够垂直方向对齐。

初始化的变量其赋值操作符应当保持一致的缩进。

未初始化变量应当放在初始化变量之后。

4.6 函数声明（zakas 4.2与A.7.4）

应当先声明函数再使用。

函数名和开始圆括号之间不应当有空格。

开始和结束圆括号与函数参数之间不应当有空格。

函数参数之间应当在逗号之后留一个空格。

结束的圆括号和右边的花括号之间应当留一个空格。

右边的花括号应当同function关键字保持在同一行。

举例（推荐的写法）：

```
function doSomething(arg1, arg2) {
   return arg1 + arg2;
}
```

函数内部定义的函数应当在var语句之后立即定义。

4.7 立即执行的匿名函数（zakas 4.4与A.7.4）

匿名函数可能作为方法赋值给变量或者对象的某个property。

对立即执行的匿名函数应当在函数调用外层用圆括号包裹。

添加一对圆括号不会改变代码逻辑，但当看到函数名前左圆括号标识符后，会醒目地知道是一个立即执行函数。

4.8 禁止调用可执行JavaScript字符串的函数（zakas 4.6.1）

为了避免跨站脚本攻击(Cross-Site Scripting：XSS)，禁止可执行字符串的函数，例如eval( )、Function( )、setTimeout( )和setInterval( )等。

只允许在将Ajax的返回值转换为JavaScript值—JSON解析的场景下使用eval( )，其他情况下禁止使用eval( )。可参见RFC7159-The JavaScript Object Notation (JSON) Data Interchange Format (Obsoletes RFC4627, RFC7158)。

禁止使用Function( )。

禁止给setTimeout( )和setInterval( )传入字符串参数。

4.9 严格模式（zakas 4.5与A.7.7）

严格模式（”use strict”）应当仅限在函数内部使用，不能再全局使用。

4.10 赋值（zakas A.7.8）

当给变量赋值时，如果右侧是含有比较语句的表达式，需要用圆括号包裹。

举例（推荐的写法）：

```
var flag = (i < count);
```

4.11 等号运算符（zakas 4.6与A.7.9）

使用===（严格相等）和!==（严格不相等）来代替==（相等）和!=（不等）以避免弱类型转换错误。

4.12 三元运算符（zakas A.7.10）

三元运算符应当仅仅用在条件赋值语句中，而不要作为if语句的替代品。

举例（推荐的写法）：

```
var value = condition ? value1 : value2;
```

5. 编程风格之语句和表达式

5.1 块语句使用花括号（zakas 3与A.7.11）

不论块语句（block statement）包含单行还是多行代码，都应当总是使用花括号。

块语句的关键词包括：if、for、while、do…while、switch、try…catch…finally

关键词后面应当紧跟一个空格，左圆括号应当在空格之后。

右圆括号后面应当紧跟一个空格，开始的大括号（即左花括号）应当在块语句中第一行的末尾，而不是放在首行的下一行。

括起来的语句应当较第一行缩进一个层级。

5.2 if语句（zakas 3.2与A.7.11）

举例（推荐的if语句写法）：

```
if (condition) {
    statements;
} else if (condition2) {
    statements;
} else {
    statements;
}
```

5.3 for语句（zakas 3.5与A.7.11）

for语句的初始化部分不应当有变量声明。

举例（推荐的for语句写法）：

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

5.4 for in语句（zakas 3.6与A.7.11）

for…in语句用于遍历对象属性，推荐总是使用object.hasOwnProperty(property)来过滤出实例属性。除非想要查找从原型继承来的属性，此时应当补充注释。

禁止使用for…in循环来遍历数组成员。

举例（推荐的for…in语句写法）：

```
var prop;

for (prop in object) {
    if (object.hasOwnProperty(prop)) {
        console.log(“property name is” + prop);
        console.log(“property value is” + object[prop]);
    }
}
```

5.5 while语句（zakas 3.2与A.7.11）

举例（推荐的while语句写法）：

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

5.6 do while语句（zakas 3.2与A.7.11）

举例（推荐的do…while语句写法）：

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

5.7 switch语句（zakas 3.3与A.7.11）

switch下的每一个case都应当保持一层缩进。

除第一个case之外，包括default在内的每一个case都应当在之前保持一个空行。

每一组语句（除了default）都应当以break、return、throw结尾，或者用一行注释表示跳过。

举例（推荐的switch语句写法）：

```
switch (expression) {
  case expression1:
    /* falls through */

  case expression2:
    statements;
    break;

  case expression3:
    return true;

  default:
    throw new Error("This shouldn't happen.");;
}
```

如果一个switch语句中不处理default情况，也不应当省略default语句，同时补充一句注释说明这个default语句什么也不做（do nothing）。

举例（推荐的switch语句写法）：

```
switch (expression) {
  case expression1:
    statements;
    break;

  default:
    // do nothing
}
```

5.8 with语句（zakas 3.4与A.7.13）

禁止使用with语句。

6. 编程实践之UI层的松耦合（zakas 5）

Web开发中用户界面（UI）由三个彼此隔离又相互作用的层来定义：

HTML：定义页面的数据和语义。

CSS：给页面添加样式，创建视觉特征。

JavaScript：给页面添加行为，使其更具交互性。

UI层松耦合的几个原则：

将CSS从HTML代码中抽离：将CSS代码放在独立文件中。

将JavaScript代码从CSS中抽离

将CSS从JavaScript代码中抽离

将JavaScript代码从HTML中抽离：将JavaScript代码放在独立文件中。

将HTML从JavaScript代码中抽离

7. 编程实践之全局变量

7.1 避免全局变量（zakas 6.1）

需要尽量避免创建全局变量和全局函数。

7.2 避免意外创建的全局变量（zakas 6.2）

当给一个未被var语句声明过的变量赋值时，JavaScript会自动创建一个全局变量。

举例：下面无意中引入了一个全局变量：

```
function doSomething( ) {
   var count = 10;   // 这里第一个变量后不小心输入了分号而不是逗号，导致title变量成为全局变量。
       title = “”hello”;
}
```

可使用JSHint等工具来报警。

7.3 每一个类库中使用单全局变量模式（zakas 6.3）

jQuery类库中使用$和jQuery两个全局对象。

可采用命名空间来对全局对象做功能分组。JavaScript中可使用对象来创建命名空间。

举例下面创建两个命名空间：

```
var ZakasBooks = {};
```

```
// 一个命名空间
ZakasBooks.MaintainableJS = {};

// 另一个命名空间
ZakasBooks.High PerformanceJS = {};
```

8. 编程实践之类型检查

8.1 避免全局变量（zakas 6.1）

需要尽量避免创建全局变量和全局函数。

9. 编程实践之配置数据独立

9.1 避免全局变量（zakas 6.1）

需要尽量避免创建全局变量和全局函数。

10. 编程实践之抛出自定义错误

10.1 避免全局变量（zakas 6.1）

需要尽量避免创建全局变量和全局函数。

11. 编程实践之不是自己的对象不要修改

11.1 避免全局变量（zakas 6.1）

需要尽量避免创建全局变量和全局函数。

12. 附录

列出本文档的附件资料（可裁剪）

……