JavaScript代码风格

内容列表

缩进

- 将tab设为4个空格。

```
// bad
function() {
··var name;
}

// bad
function() {
·var name;
}

// good
function() {
····var name;
}
```

- `if`、 `else`、 `switch`、 `case`、 `for`、 `while` 和 `:` 后1个空格。

```
// bad
if(a > b) {

}
// good
if (a > b) {

}

// bad
for(var i = 0; i < 5; i++) {
    // ...
}
// good
for (var i = 0; i < 5; i++) {
    // ...
}
```

- 赋值语句两边各1个空格

```
// bad
name='John';

// good
name = 'John';
```

- 两元运算符（`==`, `>`, `<`等）两边各1个空格。

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

```
}

// good
if (a == b) {

}
```

- 函数参数之间1个空格。

```
// bad
function ajax(url,param,success) {
    // ...
}

// good
function ajax(url, param, success) {
    // ...
}
```

- 左大括号前1个空格。

```
// bad
function ajax(url, param, success){
    // ...
}
if (a > b){

}

// good
function ajax(url, param, success) {
    // ...
}
if (a > b) {

}
```

- 一行一条语句，语句行之间4个空格缩进，不使用Tab（可将编辑器的tab设置为4个空格）。

```
// bad
var name = 'John'; var obj = {};
if (name = 'Lily') { login(name); }

// good
var name = 'John';
var obj = {};
if (name = 'Lily') {
    login(name);
}
```

- 一元运算符（`++`，`--`）前后不加空格。

```
// bad
++ sum;
total --;

// good
++sum;
total--;
```

分号

- 语句始终使用分号结尾，`for`、`function`、`if`、`switch`、`try`、`while`除外，不要依赖于引擎隐式插入。

```
// bad
function fn() {
    // ...
};
```

```
// good
function fn() {
    // ...
}

// bad
for (var i = 0; i < 5; i++) {
    // ...
};
// good
for (var i = 0; i < 5; i++) {
    // ...
}
```

逗号

- 不要将逗号放前面。

```
// bad
var hero = {
    firstName: 'Bob'
  , lastName: 'Parr'
  , heroName: 'Mr. Incredible'
  , superPower: 'strength'
};

// good
var hero = {
  firstName: 'Bob',
  lastName: 'Parr',
  heroName: 'Mr. Incredible',
  superPower: 'strength'
};
```

引号

- 引号多用来定义字符串，始终使用单引号。

```
// bad
var name = "John";

// good
var name = 'John';
```

- HTML属性使用双引号，即单引号在外层，双引号在内层。

```
// bad
var html = "<a href='http://www.jd.com'>京东</a>";

// good
var html = '<a href="http://www.jd.com">京东</a>';
```

大括号

- 起首的大括号跟在关键字的后面，且大括号前留有一个空格。

```
// bad
if (a > b)
{
    // ...
}
// good
if (a > b) {
    // ...
}

// bad
function test(){
    console.log('test');
```

```
}
// good
function test() {
    console.log('test');
}

// bad
dog.set('attr',{
    age: '1 year',
    breed: 'Bernese Mountain Dog'
});
// good
dog.set('attr', {
    age: '1 year',
    breed: 'Bernese Mountain Dog'
});
```

中括号

- 定义数组，成员以逗号分隔，逗号后加1个空格。

```
// bad
var arr = [1,2,3,4];

// good
var arr = [1, 2, 3, 4];
```

- 取对象属性用点号，非标准属性才使用中括号。

```
var obj = {name: 'John'};

// bad
  var name = obj['name'];

// good
var name = obj.name;
```

小括号

- 作为函数调用时函数名与左小括号之间没有空格。

```
// bad
func ();

// good
func();
```

- 作为强制运算符时左括号前加1个空格。

```
// bad
return(x + y);

// good
return (x + y);
```

对象属性

- 不加引号（除非一些特殊关键字或非合法标识符）。

```
// bad
var obj = {
    'name': 'John',
    'age': 30
}

// good
var obj = {
    name: 'John',
    age: 30
}
```

变量声明

- 驼峰式：变量、属性、方法名开始的第一个单词小写，之后的单词首字母大写。

```
// bad
var nickname = 'John';
var nick_name = 'John';

// good
var nickName = 'John';
```

- 避免单个字符名，让你的变量名有描述意义（计数器除外）。

```
// bad
function q() {
    // ...stuff...
}

// good
function query() {
    // ..stuff..
}
```

- 常量名全部用大写字母。

```
// bad
var pi = 3.1415926;

// good
var PI = 3.1415926;
```

- 类（构造器）名首字母大写，用名词。

```
// bad
function person(name, age) {
    this.name = name;
    this.age = age;
}

// good
function Person(name, age) {
    this.name = name;
    this.age = age;
}
```

- 多个变量，使用多个 `var` 关键字，不使用逗号分隔。

```
// bad
var name = 'John',
    age = 30,
    gender = 'male';

// good
var name = 'John';
var age = 30;
var gender = 'male';
```

- 前缀参考

前缀	类型	示例
is, can has	Boolean	isPass
get	Getter	getName
set	Setter	setName
i, j, k	iterator	
el	HTMLElement	elNav

$	jQuery Object	$nav

注释

- 在单行注释符后留一个空格。

```
// bad
//this is comment

// good
// this is comment
```

- 在多行注释的结束符前留一个空格，使星号对齐。

```
// bad
/*
  this is comment
*/

// good
/*
   this is comment
 */
```

- 不要把注释写在多行注释的开始符、结束符所在行。

```
// bad
/* start

end */
/*
here is line 1
here is line 2
 */

// good
/*
   start
   end
 */
/*
   here is line 1
   here is line 2
 */
```

- 如果某段代码有功能未实现或有待完善，必须添加“TODO”标记，“TODO”前后应留一个空格。

```
// bad
// 未处理IE的兼容性
function setOpacity(node, val) {
    node.style.opacity = val;
}

// good
// TODO 未处理IE6-8的兼容性
function setOpacity(node, val) {
    node.style.opacity = val;
}
```

- 文档注释会以预定格式出现在API文档中。以“/**开头，“/”结束，其间的每一行均以“*”开头且与开始符的第一个“*”对齐，注释内容与“*”间留一个空格。

```
// bad
/*
 * comment
 */

// good
```

```
/**
 * comment
 */
```

- 文档注释必须包含一个或多个注释标签。

```
// bad
/**
 * 模块说明
 */

// good
/**
 * 模块说明
 * @module 模块名
 */

/**
 * Core模块提供最基础的接口
 * @module Core
 */
```

- class必须搭配@constructor或@static使用，分别标记非静态类与静态类。

```
// bad
/**
 * 组件基类
 * @class Component
 */
function Componetn(nodes) {
  // ...
}

// good
/**
 * 组件基类
 * @class Component
 * @constructor
 * @param {ArrayLike<Element>} nodes 初始化节点
 */
function Componetn(nodes) {
    // ...
}
```

- 声明函数或类方法，没有指定@for时，表示此函数为全局或顶层函数。当函数为静态函数时，必须添加@static；当函数有参数时，必须使用@param；当函数有返回值时，必须使用@return。

```
// bad
/**
 * 返回当前集合中指定位置的元素
 * @method
 */
Component.prototype.getNode = function(i) {
    // ...
}

// good
/**
 * 返回当前集合中指定位置的元素
 * @method
 * @for Component
 * @param {Number} 位置下标。如果为负数，则从集合的最后一个元素开始倒数
 * @return {Element} 指定元素
 */
Component.prototype.getNode = function(i) {
    // ...
}
```

- 声明对象属性，@property

```
// bad
/**
 * id 元素id
 * classNames 样式
 */
var htmlOptions = {
    id:null,
    classNames: null
}
htmlOptions.id = "123";
htmlOptions.classNames = "arrow area";

// good
/**
 * @property {IDString} id 元素id
 * @property {ClassString} classNames 样式
 */
var htmlOptions = {
    id:null,
    classNames: null
}
htmlOptions.id = "123";
htmlOptions.classNames = "arrow area";
```

- 文件注释位于文件的最前面，应包括文件的以下信息：概要说明及版本（必须）、项目地址（开源组件必须）、版权声明（必须）、开源协议（开源组件必须）、版本号（必须）、修改时间（必须）。

```
// bad
/*!
 * Z.js Javascript Library
 */

// good
/*!
 * Z.js Javascript Library v1.0.0
 * @snandy - (2013-03-15)
 * http://github.com/snandy/z.js
 * Released under MIT license
 */
```

License

The MIT License (MIT)

Copyright (c) 2015 snandy

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

返回列表