JavaScript编码规范

1. 写在前面

> 本次的软件工程实践，前端由3个人共同开发，为了保证 javascript 的可读性，提高协作效率。故做以下编码约定。
>
> 本规范还未经过实践的检验，后期还会对本规范的条目进行增删，以适应实际开发

2 代码风格

2.1 缩进

- 使用 `4` 个空格做为一个缩进层级，不允许使用 `2` 个空格 或 `tab` 字符。
- `switch` 下的 `case` 和 `default` 必须增加一个缩进层级。

  示例：

```
// good
switch (variable) {

    case '1':
        // do...
        break;

    case '2':
        // do...
        break;

    default:
        // do...

}
```

2.2 空格

- 二元运算符两侧必须有一个空格，一元运算符与操作对象之间不允许有空格。

  示例：

```
var a = !arr.length;
a++;
a = b + c;
```

- 用作代码块起始的左花括号 `{` 前必须有一个空格。
- `if / else / for / while / function / switch / do / try / catch / finally` 关键字后，必须有一个空格。

  示例：

```
// good
if (condition) {
}

while (condition) {
}

(function () {
})();
```

- 函数声明、具名函数表达式、函数调用中，函数名和 `(` 之间不允许有空格。

  示例：

```
// good
function funcName() {
}

var funcName = function funcName() {
};

funcName();
```

- 单行声明的数组与对象，如果包含元素，`{}` 和 `[]` 内紧贴括号部分不允许包含空格。
- 行尾不得有多余的空格。

2.3 换行

- 每个独立语句结束后必须换行。
- 每行不得超过 `120` 个字符。
- 运算符处换行时，运算符必须在新行的行首。

  示例：

```
// good
if (user.isAuthenticated()
    && user.isInRole('admin')
    && user.hasAuthority('add-admin')
    || user.hasAuthority('delete-admin')
) {
    // Code
}

var result = number1 + number2 + number3
    + number4 + number5;
```

- 在函数声明、函数表达式、函数调用、对象创建、数组创建、for语句等场景中，不允许在 `,` 或 `;` 前换行。
- 在语句的行长度超过 `120` 时，根据逻辑条件合理缩进。

2.4 语句

- 不得省略语句结束的分号。
- 在 `if / else / for / do / while` 语句中，即使只有一行，也不得省略块 `{...}`。

  示例：

```
// good
if (condition) {
    callFunc();
}
```

2.5 命名

- `变量` 、`函数` 、 `参数` 均使用 `Camel命名法`。
- `常量` 使用 `全部字母大写，单词间下划线分隔` 的命名方式。
- `类` 使用 `Pascal命名法`。

  示例：

```
function TextNode(options) {
}
```

- 类的 `方法 / 属性` 使用 `Camel命名法`。
- `命名空间` 使用 `Camel命名法`。
- `类名` 使用 `名词`。
- `函数名` 使用 `动宾短语`。
- `boolean` 类型的变量使用 `is` 或 `has` 开头。

2.4 注释

- 单行注释

  必须独占一行。`//` 后跟一个空格，缩进与下一行被注释说明的代码一致。
- 多行注释

  避免使用 `/*...*/` 这样的多行注释。有多行注释内容时，使用多个单行注释。
- 文档化注释

  为了便于代码阅读和自文档化，以下内容必须包含以 `/**...*/` 形式的块注释中。
- 类注释

  使用 `@class` 标记类或构造函数。

  使用 `@extends` 标记类的继承信息。
- 函数/方法注释

  函数/方法注释包含函数说明，有参数和返回值时使用注释标识。

  参数和返回值注释包含类型信息和说明。

3 语言特性

3.1 变量

- 变量在使用前必须通过 `var` 定义。
- 每个 `var` 只能声明一个变量。
- 变量必须 `即用即声明`，不得在函数或其它形式的代码块起始位置统一声明所有变量。

  示例：

```
// good
function kv2List(source) {
    var list = [];

    for (var key in source) {
        if (source.hasOwnProperty(key)) {
            var item = {
                k: key,
                v: source[key]
            };
            list.push(item);
        }
    }

    return list;
}
```

3.2 条件

- 在 Equality Expression 中使用类型严格的 `===`。仅当判断 null 或 undefined 时，允许使用 `== null`。

  使用 === 可以避免等于判断中隐式的类型转换。

  示例：

```
// good
if (age === 30) {
    // ......
}
```

- 对于相同变量或表达式的多值条件，用 `switch` 代替 `if`。

3.3 对象

- 使用 `{}` 创建新 `Object`。
- 对象创建时，如果一个对象的所有 `属性` 均可以不添加引号，则所有 `属性` 不得添加引号。

  示例：

```
var info = {
    name: 'someone',
    age: 28
};
```

3.4 其他

- 使用 `[]` 创建新数组，除非想要创建的是指定长度的数组。
- 一个函数的长度控制在 `50` 行以内。
- 一个函数的参数控制在 `6` 个以内。