前端开发规范

> *推荐阅读：* `airbnb` *编码规范*
>
> *中文版*

一、通用规范

1.命名规范

- 项目命名：全部采用小写方式， 以下划线分隔。例如`my_project_name`
- 目录命名：全部采用小写方式，完整英文命名的用复数形式，缩写用单数形式。例如：
  `scripts, js, styles, css, images, img, data_models, modules, lego_modules, pages`
- 文件命名：全部采用小写方式,不包含空格，例如
  `order_model.js, order_detail.css, goods_detail.html`
- 模块命名：帕斯卡命名法 - 首字母大写。例如：
  `StudentInfo、UserInfo、ProductInfo`
- 变量或函数名：驼峰命名法 - 首字母小写。例如：
  `studentInfo、userInfo、productInfo`

> ***变量命名推荐：*** *`类型/名词 + 对象`描述的方式，如果没有明确类型，可以用名词作为前缀。*
>
> | ***类型*** | ***小写字母*** |
> | --- | --- |
> | *array* | *a* |
> | *boolean* | *b* |
> | *function* | *f* |
> | *int* | *i* |

object	*o*
regular	*r*
string	*s*

```
var tableTitle = "LoginTable";      //推荐

var getTitle = "LoginTable";     //不推荐
```

函数名推荐：`动词+对象` *描述的方式，构造函数使用大驼峰命名法。*

动词	***含义***	***返回值***
can	*判断是否可执行某个动作（权限）*	*函数返回一个布尔值*
has	*判断是否含有某个值*	*函数返回一个布尔值*
is	*判断是否为某个值*	*函数返回一个布尔值*
get	*获取某个值*	*函数返回一个非布尔*
set	*设置某个值*	*无返回值，返回是否*

```
//是否可阅读
function canRead(){
    return true;
}

//获取姓名
function getName{
    return this.name
}
```

- 常量命名：全部大写字母，下划线分隔。例如：`const MAX_COUNT=10;`
- 类：类名首字母大写的帕斯卡命名规范。例：`class Book {}`
  - 类的公有属性和方法采用变量命名一样的规则（驼峰）；
  - 类的私有属性和方法前缀为下划线 `_` 后跟驼峰命名规则；

2.注释规范

- 单行注释
  - 单独一行：//(双斜线)与注释文字之间保留一个空格
  - 在代码后面添加注释：//(双斜线)与代码之间保留一个空格，并且//(双斜线)与注释文字之间保留一个空格。
  - 注释代码：//(双斜线)与代码之间保留一个空格。

```
// 单独一行
setTitle();

var maxCount = 10; // 在代码后面注释

// setName(); // 注释代码
```

- 多行注释
  - 若开始(/*)和结束(*/)都在一行，推荐采用单行注释
  - 若至少三行注释时，第一行为/*，最后行为*/，其他行以*开始，并且注释文字与*保留一个空格。

```
/*
* 代码执行到这里后会调用setTitle()函数
* setTitle(): 设置title的值
*/
setTitle();
```

  - **函数注释**：函数的注释也是多行，但比较特殊，需要包含说明信息。语法如下：

```
/**
* 函数说明
* @关键字
*/
```

常用注释关键字：（example 指的是函数用法）

注释名	语法	含义	示例
@param	@param 参数名（参数类型）描述信息	描述参数的信息	@param 传入名称

@return	@return {返回类型} 描述信息	描述返回值的信息	@return true:可执
@author	@author 作者信息 [附属信息：如邮箱、日期]	描述此函数作者的信息	@author 2015/07/
@version	@version XX.XX.XX	描述此函数的版本号	@version
@example	@example 示例代码	@example setTitle('测试')	如下

```
  /**
   - 合并Grid的行
   - @param grid {Ext.Grid.Panel} 需要合并的Grid
   - @param cols {Array} 需要合并列的Index(序号)数组；从0开始计数，序号也包含。
   - @param isAllSome {Boolean} ：是否2个tr的cols必须完成一样才能进行合并。tr
ue：完成一样；false(默认)：不完全一样
   - @return void
   - @author polk6 2015/07/21
   - @example
   - _________________                              _________________
   - |  年龄 |  姓名 |                            |  年龄 |  姓名 |
   - -----------------      mergeCells(grid,[0])    -----------------
   - |  18   |  张三 |              =>            |       |  张三 |
   - -----------------                            -  18   ---------
   - |  18   |  王五 |                            |       |  王五 |
   - -----------------                             -----------------
  */
  function mergeCells(grid, cols, isAllSome) {
      // Do Something
  }
```

二、HTML 规范

- 文档类型声明 `<!DOCTYPE html>`
- JavaScript 脚本文件放在 `</body>` 标签前
- 使用语义化标签，结构类似：

```
<!DOCTYPE html>
<header>
<h1>
      ...
  </h1>
```

```
</header>
<nav></nav>
<main>
<section>
      <header></header>
    <article>
        <header></header>
          <aside></aside>
          <footer></footer>
      </article>
      <footer></footer>
  </section>
  <section></section>
</main>
<footer></footer>
```

- 图片标签的 alt 属性不能为空
- 视觉设计内容建议使用伪属性 `:before` 或 `:after` 来模拟，而不是添加新的 HTML 标签：

```
<!-- 不推荐使用 -->
<!-- html 代码:-->
<span class="text-box">
<span class="square"></span>
See the square next to me?
</span>
<!--css 代码:-->
.text-box > .square {
display: inline-block;
width: 1rem;
height: 1rem;
background-color: red;
}
<!-- 推荐使用 -->
<!-- html 代码:-->
<span class="text-box">
See the square next to me?
</span>
<!--css 代码:-->
.text-box:before {
content: "";
display: inline-block;
width: 1rem;
height: 1rem;
background-color: red;
}
```

三、CSS 规范

1.命名规范

- 全部小写，样式名一般以用途或目的命名，不建议使用 ID 作为样式名
- 命名中有多个单词推荐使用短号 `-` 连接

2.选择器

- 推荐多使用直接子选择器 `>`
- 避免减少标签名的出现

3.属性

- 推荐使用缩略方式编写属性，如`font-size: 10px,font-family:Arial` 缩写成 `font:10px Arial`
- 数值为 0 时可省略单位，如 `margin:0px;` 建议为 `margin:0`

四、JS 规范

1.缩进：2个空格

2.单行长度：不要超过120，长字符串拼接用加号。

3.分号：以下几种情况后需加分号：

- 变量声明
- 表达式
- return
- throw
- break
- continue
- do-while

```
/* var declaration */
var x = 1;

/* expression statement */
x++;
```

```
/* do-while */
do {
    x++;
} while (x < 10);
```

4.空格

以下几种情况不需要空格：

- 对象的属性名后

```
// not good
var a = {
    b :1
};
// good
var a = {
    b: 1
};
```

- 无论是函数声明还是函数表达式，'('前不要空格

```
// not good
function add () {

}
// good
function add() {

}
```

- 数组的'['后和']'前

```
// not good
[ 1, 2 ]
// good
[1, 2]
```

- 对象的'{'后和'}'前

```
// not good
{ title: '对象' }
// good
{title: '对象'}
```

- 运算符'('后和')'前

```
// not good
var a = ( 1+2 )*3;

// good
var a = (1 + 2) * 3;
```

以下几种情况需要空格：

- 三元运算符'?:'前后

```
var y = x ? 1 : 2;
```

- 代码块'{'前
- 下列关键字前：else, while, catch, finally
- 下列关键字后：
  if, else, for, while, do, switch, case, try, catch, finally, with, return, typeof
- 单行注释'//'后（若单行注释和代码同行，则'//'前也需要），多行注释'*'后
- 对象的属性值前
- for循环，分号后留有一个空格，前置条件如果有多个，逗号后留一个空格

```
// not good
for(i=0;i<6;i++){
    x++;
}

// good
for (i = 0; i < 6; i++) {
    x++;
}
```

- 无论是函数声明还是函数表达式，'{'前一定要有空格

```
// not good
function test()
{
    ...
}

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

- 函数的参数之间

```
// not good
function foo(a,b,c) {

}
// good
function foo(a, b, c, d, g, j) {
    ...
}
```

5.模块

- 使用立即执行函数表达式（`IIFE`）封装全局模块，`ES6`中使用`import`(替代require) 和 `export`(替代module.exports) 命令：

```
// IIFE 模式
(function($, w, d){
'use strict';

$(function() {
  w.alert(d.querySelectorAll('div').length);
});
}(jQuery, window, document));
// ES6 模式
// profile.js
var firstName = 'Michael';
var lastName = 'Jackson';
var year = 1958;

export {firstName, lastName, year};

// main.js
```

```
import {firstName, lastName, year} from './profile.js';

function setName(element) {
element.textContent = firstName + ' ' + lastName;
}
```

- 模块只有一个输出值使用 `export default`。且 `export default` 和 `export` 不建议在同一个模块同时使用
- 如果模块默认输出一个函数，首字母小写；如果默认输出一个对象，首字母大写

```
// e.g. 1
function makeStyleGuide() {
}

export default makeStyleGuide;

// e.g. 2
const StyleGuide = {
es6: {
}
};

export default StyleGuide;
```

6.函数

- 简单的、单行的且不会复用的函数，建议使用箭头函数。立即执行函数也建议用箭头函数形式。
- 函数体内不建议获取 `arguments` 变量，统一使用扩展运算符 `...` 代替。

```
// bad
function concatenateAll() {
const args = Array.prototype.slice.call(arguments);
return args.join('');
}

// good
function concatenateAll(...args) {
return args.join('');
}
```

7.数组

- 使用扩展运算符实现拷贝（浅拷贝）

```
// bad
const len = items.length;
const itemsCopy = [];
let i;

for (i = 0; i < len; i++) {
itemsCopy[i] = items[i];
}

// good
const itemsCopy = [...items];
```

- 使用 `Set` 数据类型给数组去重 `[...new Set(array)]`
- 使用 `Array.from` 转化类对象

```
const foo = document.querySelectorAll('.foo');
const nodes = Array.from(foo);
```

8.对象

- 使用 `assign` 添加新属性

```
const a = {};
Object.assign(a, { x: 3 });
```

- 使用属性表达式定义动态属性名

```
// bad
const obj = {
id: 5,
name: 'San Francisco',
};
obj[getKey('enabled')] = true;

// good
const obj = {
id: 5,
```

```
name: 'San Francisco',
[getKey('enabled')]: true,
};
```

- 对象属性名不需要加引号

```
// not good
var a = {
    'b': 1
};
// good
var a = {
    b: 1
};
```

9.字符串

- 静态字符串使用单引号或者反引号，动态字符串配合模板字符使用反引号；

```
// bad
const a = "foobar";
const b = 'foo' + a + 'bar';

// acceptable
const c = `foobar`;

// good
const a = 'foobar';
const b = `foo${a}bar`;
```

10. 使用 ESLint 校验代码

> - *安装`ESLint`：`npm i -g eslint` 使用VScode编辑器开发，安装ESlint扩展插件 在项目根目录中新建`.eslintrc`文件，设置规则rules，搭配VScode的ESlint插件使用，按保存可以自动调整代码*

五、React 规范

> `React`[规范]

> 中文版

六、Vue 规范

> 官宣 vue风格指南

本文其他参考内容：

> 前端开发规范
>
> es6 编程风格