XU

Nuesiri |

JavaScript代码规范

版本追踪

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档版本 | 日期 | 作者 | 操作 | 描述 |
| V0.5 | 2018-03-20 | 徐健 | 创建 | 创建文档并搭建基本框架。 |
| V1.0 | 2018-11-11 | 徐健 | 增加 | 添加预处理框架章节 |
| V1.1 | 2018-11-23 | 徐健 | 增加 | ES6及vue语法规范 |

## 命名规范

* 变量名主体使用小写驼峰命名,后面要有数据类型标识（包括基本数据类型和引用类型），且同样遵循小写驼峰命名规范。

数据类型声明：

|  |  |
| --- | --- |
| **类型** | **标识** |
| 对象（Object） | Obj |
| 字符串（String） | Str |
| 数组（Array） | Arr |
| 数字（Number） | Num |
| 布尔（Boolean） | Bool |
| 浮点（Float） | Float |
| 日期（Date） | Date |

注：声明浮点类型必须要加注释，且避免直接用浮点类型进行科学计算。

1. **var** personObj = **new** Object();
2. person.name = "Bruce";
3. person.age = 21;
4. person.family = ["lida","lier","wangwu"];
5. person.say = **function**(){
6. alert(**this**.name);
7. }
8. **var** ageNum = 1233;
9. **var** personcash **= 19.2；//用于计算员工的每月结算金额**
10. **var** neuPersonObj ={
11. name: "lisi",
12. age: 21,
13. family: ["lida","lier","wangwu"],
14. say: **function**(){
15. alert(**this**.name);
16. }
17. };

* 函数名使用小写驼峰写法，类名使用大写驼峰写法。

1. **function** myFunction()
2. {
3. alert("Hello World!");
4. }
5. **class** Person {
6. constructor(name) {
7. **this**.name = name;
8. }
9. say () {
10. console.log("say hi");
11. }
12. };

**备注：ES6中的类在解析时不会变量声明提前，必须在声明后调用。这里建议使用ES5原型链的方式来表达类的继承。**

1. **function** Animal(name) {
2. **this**.name = name
3. }
5. Animal.prototype.getName = **function**() {
6. console.log(**this**.name)
7. }
9. **var** animal1 = **new** Animal('Kate')
10. **var** animal2 = **new** Animal('Lucy')

**注：这里的Animal表示类名，所以使用大写驼峰写法。**

* 私有成员必须以下划线开头。

1. **function** Car(){
2. **var** \_wheel = 3;
3. **this**.wheel = 4;
4. **this**.getPrivateVal = **function**(){
5. **return** \_wheel;
6. }
7. }
8. **var** car1 = **new** car();
9. car1.wheel = 4;
10. console.log(car1.getPrivateVal());

* 常量名必须使用全部大写的下划线命名法。

1. **var** DATABASE\_URL = '172.0.0.1:8080';

* 如果使用严格模式，出了常规保留字以外，下列保留字也禁止使用（ES5）。

Implements、package、public、interface、private、static、let、proteceted、yield

## 注释规范

* 文件注释

1. /\*
2. \* @作者: 徐健
3. \* @日期: 2018-11-20 09:17:00
4. \* @最后修改:   徐健
5. \* @最后修改时间: 2018-11-20 09:17:00
6. \*/

**注：如果有页面级别的插件引用的话可以在此说明**

* 单行注释
* 多行注释
* Class声明

1. /\*\*
2. \* 节点集合类
3. \* @class NodeList
4. \* @constructor
5. \* @param {ArrayLike<Element>} nodes 初始化节点
6. \*/

* 方法声明

1. /\*\*
2. \* @description
3. \* @author bruce
4. \* @date 2018-11-19
5. \* @修改人 jim
6. \* @修改时间 2018-11-20
7. \*/
8. function add(){
9. console.log('bruce');
11. };

## 关于VUE框架

* 目录结构遵循Vue-cli脚手架的结构基础划分

1. ├── index.html                      入口页面
2. ├── build                           构建脚本目录
3. │   ├── build-server.js                 运行本地构建服务器，可以访问构后的页面
4. │   ├── build.js                        生产环境构建脚本
5. │   ├── dev-client.js   服务器热重载脚本，主要用来实现开发阶段的页面自动刷新
6. │   ├── dev-server.js                   运行本地开发服务器
7. │   ├── utils.js                        构建相关工具方法
8. │   ├── webpack.base.conf.js            wabpack基础配置
9. │   ├── webpack.dev.conf.js             wabpack开发环境配置
10. │   └── webpack.prod.conf.js            wabpack生产环境配置
11. ├── config                          项目配置
12. │   ├── dev.env.js                      开发环境变量
13. │   ├── index.js                        项目配置文件
14. │   ├── prod.env.js                     生产环境变量
15. │   └── test.env.js                     测试环境变量
16. ├── mock                            mock数据目录
17. │   └── hello.js
18. ├── package.json        npm包配置文件，里面定义了项目的npm脚本，依赖包等信息
19. ├── src                             项目源码目录
20. │   ├── main.js                         入口js文件
21. │   ├── App.vue                         根组件
22. │   ├── components                      公共组件目录
23. │   │   └── title.vue
24. │   ├── assets                          资源目录，这里的资源会被wabpack构建
25. │   │   ├── css                         公共样式文件目录
26. │   │   ├── js                          公共js文件目录
27. │   │   └── img                      图片存放目录
28. │   ├── routes                          前端路由
29. │   │   └── index.js
30. │   ├── store                           应用级数据（state）
31. │   │   └── index.js
32. │   └── views                           页面目录
33. │       ├── hello.vue
34. │       └── notfound.vue
35. ├── static                          纯静态资源，不会被wabpack构建。
36. └── test                            测试文件目录（unit&e2e）
37. └── unit                            单元测试
38. ├── index.js                        入口脚本
39. ├── karma.conf.js                   karma配置文件
40. └── specs                           单测case目录
41. └── Hello.spec.js

* 不要使用this.$parent
* 无需将 this 赋值给 component 变量
* 页面级css要声明语法框架（遵循css规范）

1. **<style** lang="less" scoped**>**
2. **</style>**

* vue文件方法声明顺序

1. - components
2. - props
3. - data
4. - created
5. - mounted
6. - activited
7. - update
8. - beforeRouteUpdate
9. - metods
10. - filter
11. - computed
12. - watch
13. – 其它方法声明

## 其它注意事项

* 写公共API(如工具包)时，在开头尽量要加分号“;”。
* 禁止简写三元以下运算符的简写。

反例：

1. // 这是反例
2. **if** (**true**)
3. alert(name);
4. console.log(name);
5. // 这是反例
6. **if** (**true**)
7. alert(name);
8. console.log(name);

* 使用局部变量来缓存dom指向和作用域链较深的对象，不要使用self

1. // 缓存对象
2. **var** getComment = **function**() {
3. **var** dom = $("#common-container"),               // 缓存dom
5. }

8. //当需要缓存this时必须使用\_self变量进行缓存
9. // 缓存this
10. **function** Row(name) {
11. **var** \_self = **this**;
13. \_self.name = name;
14. $(".row").click(**function**() {
15. \_self.getName();
16. });
17. }

* 声明变量时要使用var（let、const），不允许直接声明
* 写自执行函数的时候前面要加“；”每条语句结尾要加“；“（import除外）

1. a = b        // 赋值
2. (**function**(){
3. //....
4. })()         // 自执行函数

* 关于ES6定义变量使用let ,定义常量使用const
* 调试信息console.log() debugger 使用完及时删除
* 不要尝试使用双击做交互事件