区域卫生研发部研发规范

JS

2018年06月

修改历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **修改人** | **修改日期** | **修改内容** | **审批人** |
| 0.1 | 潘豪格 | 2018/06/07 | 创建 |  |
| 0.2 | 刘建锋 | 2018/07/16 | 初版 |  |
| 0.3 | 刘建锋 | 2018/08/05 | 修正 |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[**1.** **命名规范** 3](#_Toc520813316)

[1.1. 总则 3](#_Toc520813317)

[1.2. 变量 4](#_Toc520813318)

[1.3. 常量 4](#_Toc520813319)

[1.4. 方法/函数 4](#_Toc520813320)

[1.5. JS文件 6](#_Toc520813321)

[**2.** **注释规范** 7](#_Toc520813322)

[2.1. 公共组件规范 7](#_Toc520813323)

[2.2. 模块、函数及方法注释规范 7](#_Toc520813324)

[2.3. 代码间注释规范 9](#_Toc520813325)

[**3.** **JS代码规范** 9](#_Toc520813326)

[3.1. 代码格式 9](#_Toc520813327)

[3.2. 代码优化 10](#_Toc520813328)

[**4.** **其它原则** 14](#_Toc520813329)

1. **命名规范**
   1. 总则

* 【强制】无论是变量,类,还是其他对象的命名,都应该采用意义明确的命名,根据命名规范命名，以增加程序的可读性和可维护性。
* 【强制】区分大小写，**提示：**JavaScript 对大小写敏感
* 【强制】名称，全部用拼音

正例：病人信息：BingRenXX;

正例：获取病人信息:GetBingRenXX()

* 1. 变量

使用场景：局部变量、全局变量、参数

* 【强制】名称应该能够标识变量的特征、用途；
* 【强制】名称前两个中文字全拼+后续中文拼音首字母;若拼音组合一样，则将第三个字改为全拼，若第三个字全拼仍无法区分，则第四个字改为全拼，依次类推。
* 【强制】字母全部小写；

正例：个人编号命名为gerenbh: 如： var gerenbh=“2018072500001”

反例：var a,随意命名方式

反例：var ABC,不允许大写

批次信息总数

正例：var picixxzs=””;

反例：var picixxcount,不允许出现拼音+英文的方式

* 1. 常量
* 【强制】名称应该能够标识常量的特征、用途；
* 【强制】名称前两个中文字全拼+后续中文拼音首字母;若拼音组合一样，则将第三个字改为全拼，若第三个字全拼仍无法区分，则第四个字改为全拼，依次类推。
* 【强制】字母全部大写；
* 【强制】以CONST\_开头；
* 【推荐】尽量将全部常量定义在某一个工程中,不要在很多工程下定义常量；

正例： var CONST\_XIANCHENGJGSJ=5 //线程间隔时间

反例： var CONST\_xcjgsj=5

* 1. 方法/函数
* 【强制】名称应该能够标识方法的特征、用途；
* 【强制】关键词 function 必须是小写
* 【强制】名称以常见动词开头+前两个中文字全拼+后续中文拼音首字母;若拼音组合一样，则将第三个字改为全拼，若第三个字全拼仍无法区分，则第四个字改为全拼，依次类推；
* 【强制】拼音首字母大写，其他小写；
* 【推荐】不要使用下划线字符（\_）

常见动词表

|  |  |  |  |
| --- | --- | --- | --- |
| 动词 | 含义 | 动词 | 含义 |
| Get | 获取 | Set | 设置 |
| Add | 增加 | Remove | 删除 |
| Create | 创建 | Destory | 移除 |
| Start | 启动 | Stop | 停止 |
| Open | 打开 | Close | 关闭 |
| Read | 读取 | Write | 写入 |
| Load | 载入 | Save | 保存 |
| Create | 创建 | Destroy | 销毁 |
| Begin | 开始 | End | 结束 |
| Backup | 备份 | Restore | 恢复 |
| Import | 导入 | Export | 导出 |
| Split | 分割 | Merge | 合并 |
| Inject | 注入 | Extract | 提取 |
| Attach | 附着 | Detach | 脱离 |
| Bind | 绑定 | Separate | 分离 |
| View | 查看 | Browse | 浏览 |
| Edit | 编辑 | Modify | 修改 |
| Select | 选取 | Mark | 标记 |
| Copy | 复制 | Paste | 粘贴 |
| Undo | 撤销 | Redo | 重做 |
| Insert | 插入 | Delete | 移除 |
| Add | 加入 | Append | 添加 |
| Clean | 清理 | Clear | 清除 |
| Index | 索引 | Sort | 排序 |
| Find | 查找 | Search | 搜索 |
| Increase | 增加 | Decrease | 减少 |
| Play | 播放 | Pause | 暂停 |
| Launch | 启动 | Run | 运行 |
| Compile | 编译 | Execute | 执行 |
| Debug | 调试 | Trace | 跟踪 |
| Observe | 观察 | Listen | 监听 |
| Build | 构建 | Publish | 发布 |
| Input | 输入 | Output | 输出 |
| Encode | 编码 | Decode | 解码 |
| Encrypt | 加密 | Decrypt | 解密 |
| Compress | 压缩 | Decompress | 解压缩 |
| Pack | 打包 | Unpack | 解包 |
| Parse | 解析 | Emit | 生成 |
| Connect | 连接 | Disconnect | 断开 |
| Send | 发送 | Receive | 接收 |
| Download | 下载 | Upload | 上传 |
| Refresh | 刷新 | Synchronize | 同步 |
| Update | 更新 | Revert | 复原 |
| Lock | 锁定 | Unlock | 解锁 |
| Checkout | 签出 | Checkin | 签入 |
| Submit | 提交 | Commit | 交付 |
| Push | 推 | Pull | 拉 |
| Expand | 展开 | Collapse | 折叠 |
| Begin | 起始 | End | 结束 |
| Start | 开始 | Finish | 完成 |
| Enter | 进入 | Exit | 退出 |
| Abort | 放弃 | Quit | 离开 |
| Obsolete | 废弃 | Depreciate | 废旧 |
| Collect | 收集 | Aggregate | 聚集 |

正例 ： 获取病人信息GetBingRenXX()

反例： getbingrenxx

* 1. JS文件
* 【强制】名称应该能够标识文件的特征、用途；
* 【强制】全小写标识，包括后缀文件名js；
* 【强制】js文件发布文件要压缩； 文件名 加上.min.js 结尾； js源文件文件名 加上.src.js 结尾；

1，压缩多余的空格和换行符；

2、删除注释，因为注释对浏览用户是无用的，删除了可以缩小文件体积；

3、把较长的变量名称和过程名称统一替换为很短的名;

正例：类BingRenXX对应的文件名应为 BingRenXX.js

* 【强制】引用js文件 后面要加上 ?version=版本标识；

1. **注释规范**
   1. 公共组件规范

* 【强制】注释需要描述功能、依赖和如何使用
* 【强制】注释内容放到组件的上部；

正例： （可以参考 jquery 源码）；

/\*\*

\*文件用途说明

\*作者姓名、联系方式

\*制作日期

\*使用方法 如：$.do();

\*\*/

* 1. 模块、函数及方法注释规范
* 【强制】注释必须描述方法的用途，使用场景；
* 【强制】有参数的必须注释各个参数中文含义；
* 【强制】有返回值的，必须描述返回值中文含义；

正例：

/\*\*

\* 函数说明

\* @关键字

\*/

**常用注释关键字：(只列出一部分，并不是全部)**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 name {String} 传入名称 |
| @return | @return {返回类型} 描述信息 | 描述返回值的信息 | @return {Boolean} true:可执行;false:不可执行 |
| @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 张三 2015/07/21 |
| @version | @version XX.XX.XX | 描述此函数的版本号 | @version 1.0.3 |
| @example | @example 示例代码 | 演示函数的使用 | @example setZhuTi('测试') |

/\*\*

\* 合并表格的行

\* @param biaoge{Ext.Grid.Panel} 需要合并的表格

\* @param lies {Array} 需要合并列的Index(序号)数组；从0开始计数，序号也包含。

\* @param issuoyou {Boolean} ：是否2个tr的cols必须完成一样才能进行合并。true：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 |

\* ----------------- MergeBiaoGeH (biaoge,[0]) -----------------

\* | 18 | 张三 | => | | 张三 |

\* ----------------- - 18 ---------

\* | 18 | 王五 | | | 王五 |

\* ----------------- -----------------

\*/

function MergeBiaoGeH(biaoge, lies, issuoyou) {

// Do Something

}

* 1. 代码间注释规范
* 【强制】单行注释必须以//方式
* 【强制】多行注释必须以/\*\*/

正例：

单行注释：//<单行注释>

// 调用了一个函数；1)单独在一行

SetZhuTi();

多行注释：/\*多行注释1

多行注释2

多行注释3

\*/

/\*

\* 代码执行到这里后会调用setZhuTi()函数

\* setZhuTi()：设置ZhuTi的值

\*/

SetZhuTi();

* 【强制】全局变量注释，每个全局变量需要写注释
* 【强制】程序流程变化注释，switch, if, while 等条件判断地方必须写注释
* 【强制】关键业务逻辑必须写注释

更多注释内容，可参考JSDOC ：<http://usejsdoc.org>

1. **JS代码规范**
   1. 代码格式

* 【强制】针对JS代码书写完成后必须使用VS默认格式化进行美化；
* 【强制】代码行与行之间不允许连续出现两行及以上空行；

反例：

function GetJuMinXX() {

var say = "hello";

var flag = 0;

if (flag == 0) {

aler("空行太多");

}

}

* 1. 代码优化
* 【推荐】一个函数、方法只做一件事，避免出现一些重复的冗余的代码功能块;
* 【推荐】逻辑判断：一个方法里面不要嵌套太多的逻辑判断（if else或者switch case），嵌套达到三层的判断就可以考虑把其中的一部分独立成新方法调用，或者使用尽快返回的方式;
* 【推荐】尽量缩短变量的存活周期；尽量不使用全局变量；尽量不使用静态变量；
* 【推荐】变量跨度：变量声明定义开始到第一次使用该变量的代码行之间的行距尽可能短；
* 【推荐】占用内存较高的变量建议手工释放；
* 【推荐】当一个方法的参数数量比较多的时候，如果可能，声明一个单独对象进行封装;
  1. Ajax 使用
* 【推荐】使用Jquery 的Ajax 开发技术；

jQuery是一个轻量级的Javascript库，兼容CSS3，还兼容各种浏览器。jQuery使用户能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。

1. **JS代码错误处理与调试**
   1. 错误处理

* 【推荐】 try-catch语句

try {

//可能会导致错误的代码

} catch (error) {

//错误发生时怎么处理

}

try {

// noneFunc 这个方法并不存在

window.noneFunc();

} catch (error) {

// 一旦报错将会跳到这里执行

console.log(error.message); // window.noneFunc is not a function

} finally{

}

// 执行不中断，这句代码会得到执行

console.log("hello");

* 【推荐】抛出错误

抛出错误使用throw关键字，对于抛出的错误类型则没有规定，可以是任意类型，而浏览器对用户抛出的错误处理也和内置的错误一致，如果没有try-catch进行包含的话，浏览器会暂停JS的执行

1. 与try-catch语句相配有一个throw操作符。在遇到throw操作符时，代码立即停止执行。仅当有try-catch语句捕获到被抛出的值时，代码才会继续执行。
2. throw new Error("something bad happened."); 或利用原型链通过继承Error来创建自定义错误类型。

* 【推荐】错误（error）事件

error事件仅支持DOM0级的监听方法，即不能通过addEventListener和removeEventListener方法来注册和移除，同时该方法也不会创建对应的event对象，而是将报错信息直接传递过来

1. 任何没有通过try-catch处理的错误都会触发window对象的error事件。
2. 任何浏览器中，onerror事件处理程序都不会创建event对象。但可接受3个参数：错误消息、错误所在的URL和行号。（只有错误消息有用）
3. 指定onerror事件处理程序，必须使用DOM0级技术。   
   window.onerror = function(message,url,line){   
   alert(message);   
   return false; //通过返回false，此函数实际上充当了整个文档的try-catch语句可捕获所有无代码处理的运行时错误。   
   };

④ 图像也支持error事件。只要图像的src特性中URL不能返回可以被识别的图像格式，就会触发error事件。此时的error事件遵循DOM格式，会返回一个以图像为目标的event对象。

* 1. 调试(debugger)规范
* 【推荐】如浏览器支持调试，可以使用 console.log()方法在调试窗口上打印相关的值；也可以设置断点: 中断代码执行便于查找问题；
* 【禁用】debugger 调试机制： 该方法需要在 js 文件中 调用端 注册 debugger ；方法； 然而 实际上 很容易忘记 在代码提交时 删除该关键字； 导致客户端浏览时 会有调试信息提示；

1. **第三方JS框架引入原则及规范** 
   1. 引入第三方框架文件目录

* 【强制】单独建立文件夹存放；
* 【强制】引入加入版本号信息；
* 【推荐】引用压缩后的版本；
  1. 避免冲突规则
* 【推荐】引用命名空间，或者第三方框架规则 如 Jquery 的 JQuery 和 $ 符号；