前端开发工具Visual Studio Code

前端环境配置node.js

前端框架vue

后端开发工具idea

后端框架spring boot

数据库连接技术JDBC

数据库sqlsever

符合web标准，结构表现行为分离，兼容性优良。页面性能方面，代码要求简洁明了有序， 尽可能的减小服务器负载，保证最快的解析速度。

项目的维护和二次开发可能是直接或间接的团队合作，所以创建易维护的代码是一个项目成功与否的关键，易维护的代码意味着具有如下特性：

阅读性好：如良好的注释和命名规范，有文档

具有一致性：看起来如同一个人编写

代码的松耦合，高度模块化：将页面内的元素视为一个个模块，相互独立，尽量避免耦合过高的代码，从html,css,js三个层面都要考虑模块化

严格按照规范编写代码

## 三、命名规范

### 1. 目的

提高代码可预测性和可维护性的方法是使用命名约定，这就意味着采用一致的方法来对变量和函数进行命名。

### 2. 变量名

变量名包括全局变量，局部变量，类变量，函数参数

### 3. 构造函数（类）命名

首字母大写，驼峰式命名。

JS中没有类，但是可以用new调用构造函数：var man = new Person();

### 4. 普通变量命名

首字母小写，驼峰式命名，匈牙利命名

如：nCheckCount 表示整形的数值

### 5. 匈牙利命名法

匈牙利命名法语法：变量名＝类型＋对象描述

* 类型指变量的类型
* 对象描述指对象名字全称或名字的一部分，要求有明确含义，命名要容易记忆容易理解。

****提示****: 虽然JavaScript变量表面上没有类型，但是JavaScript内部还是会为变量赋予相应的类型

| **JavaScript变量起名类型** | **变量命名前缀** | **举例** |
| --- | --- | --- |
| Array 数组 | a | aList，aGroup |
| Boolean 逻辑 | b | bChecked，bHasLogin |
| Function 函数 | f | fGetHtml，fInit |
| Integer 数字 | n | nPage，nTotal |
| Object 对象 | o | oButton，oDate |
| Regular Expression 正则 | r | rDomain，rEmail |
| String 字符 | s | sName，sHtml |

### 6. 其他前缀规范

可根据团队及项目需要增加

$：表示Jquery对象

例如：$Content，$Module，一种比较广泛的Jquery对象变量命名规范。

fn：表示函数

例如：fnGetName，fnSetAge；和上面函数的前缀略有不同，改用fn来代替，个人认为fn能够更好的区分普通变量和函数变量。

### 7. 例外情况

以根据项目及团队需要，设计出针对项目需要的前缀规范，从而达到团队开发协作便利的目的。

* 作用域不大临时变量可以简写，比如：str，num，bol，obj，fun，arr。
* 循环变量可以简写，比如：i，j，k等。
* 某些作为不允许修改值的变量认为是常量，全部字母都大写。例如：COPYRIGHT，PI。常量可以存在于函数中，也可以存在于全局。必须采用全大写的命名，且单词以\_分割，常量通常用于ajax请求url，和一些不会改变的数据。

### 8. 函数命名

****普通函数****：首字母小写，驼峰式命名，统一使用动词或者动词+名词形式

例如：fnGetVersion()，fnSubmitForm()，fnInit()；涉及返回逻辑值的函数可以使用is，has，contains等表示逻辑的词语代替动词，例如：fnIsObject()，fnHasClass()，fnContainsElment()。

****内部函数****：使用\_fn+动词+名词形式，内部函数必需在函数最后定义。

例如：

function fnGetNumber(nTotal) {

if (nTotal < 100) {

nTotal = 100;

}

return \_fnAdd(nTotal);

function \_fnAdd(nNumber) {

nNumber++;

return nNumber;

}

}

alert(fGetNumber(10)); //alert 101

****对象方法与事件响应函数****：对象方法命名使用fn+对象类名+动词+名词形式；

例如： fnAddressGetEmail()，

****事件响应函数****：fn+触发事件对象名+事件名或者模块名

例如：fnDivClick()，fnAddressSubmitButtonClick()

****函数方法常用的动词****：

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

check out 签出/check in 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

### 9. 变量命名例子

为什么需要这样强制定义变量前缀？正式因为javascript是弱语言造成的。在定义大量变量的时候，我们需要很明确的知道当前变量是什么属性，如果只通过普通单词，是很难区分的。

普通代码

var checked = false;

var check = function() {

return true;

}

/\*\*

some code

\*\*/

if(check) {//已经无法很确切知道这里是要用checked还是check()从而导致逻辑错误

//do some thing

}

规范后代码

var bChecked = false;

var fnCheck = function() {

return true;

}

/\*\*

some code

\*\*/

if(bChecked) {

// do some thing

}

if(fnCheck()) {

// do other thing

}

如何标明私有方法或私有属性？

var person = {

getName: function () {

return this.\_getFirst() + ' ' + this.\_getLast();

},

\_getFirst: function () {

//...

},

\_getLast: function (){

//...

} };

在这个例子中，getName()以为这这是API的一个公开的方法，而\_getFirst()和\_getLast()意味着这是一个私有函数。尽管他们都是普通的公开方法，但是使用下划线前缀的表示方法可以提醒使用person对象的用户，告诉他们这些方法在其他地方不能确保一定能够正常工作，不能直接调用。

总结：下面是一些使用下划线约定的变量

* + 使用下划线结尾来表明是私有变量，例如name\_和getElements\_()。
  + 使用一个下划线前缀来表示受保护属性，使用两个下划线前缀来表示私有属性。

## 四、编写注释

为代码编写注释是非常重要的。通常人们在深入思考一个问题时，会非常清楚这段代码的工作原理。但是当过一周后再次回到该代码时，可能会花上很长时间来回想起那段代码到底是干什么的。

公共组件维护者和各栏目WD都需要在文件头部加上注释说明：

/\*\*

\*文件用途说明

\*作者姓名、联系方式（旺旺）

\*制作日期

\*\*/

大的模块注释方法：

//================

// 代码用途

//================

小的注释；

//代码说明

注释单独一行，不要在代码后的同一行内加注释。例如：

//姓名

var name = “abc”; V

var name =”abc”; //姓名 X

****用处一****

* 在分开for循环的各个部分的分号之后：例如，for (var i = 0; i < 10; i +=1){…}
* 在for循环中初始化多个变量（i和最大值等）：for (var i = 0, max = 10; i < max; i += 1){…}
* 在限定数组项的逗号后面：var a = [1, 2, 3];
* 对象属性的逗号之后和将属性名和属性值分开的冒号之后：var o = {a: 1, b: 2};
* 分隔开函数中各个参数的逗号之后：myFunc(a, b, c)
* 在函数声明的大括号之前：function myFunc() {}
* 在匿名函数表达式之后：var myFunc = function () {};

****用初二****

空格的另外一个很好的用途是用来分隔所有的操作符和操作，这也就是意味着在 +, -, \*, =, <, >, <=, >=, ===, !==, &&, ||, += 等之后使用空格：

例子：

//大量空格，并且使用一致，是的代码可读性更好

//允许在阅读的时候不用一口气读下去

var d = 0,

a = b +1;

if ( a && b && c) {

d = a % c;

a += d;

}

//反模式

//缺少空格或空格使用不一致，使得代码比较混乱

var d= 0;

a =b+1;

if (a&& b&& c) {

d=a %c;

a+= d;

}

## 五、编写API文档

### 1. 生成API文档的步骤：

* 编写特殊格式的代码块（即一些注释块）
* 运行工具来解析代码和注释（工具如：JSDoc Toolkit和YUIDoc）
* 发布工具解析的结果，大多数情况是采用HTML格式发布（如网页版的API文档就是利用工具生成的）

****简单举例****：

/\*\*

\* 翻转一个字符串

\*

\* @param {String} 输入需要翻转的字符串

\* @return {String} 翻转后的字符串

\*\*/

var reverse = function (input) {

//...

return output;

};

****YUIDoc范例****：

完整范例：本程序由一个文件(app.js)组成，该文件仅有一个模块(myapp)。

app.js:

/\*\*

\* 我的javascript应用程序

\*

\* @module myapp

\*/

//使用命名空间来定义一个空对象

var MYAPP = {};

//定义一个包含两个方法(sum()和multi())的math\_stuff对象

/\*\*

\* @namespace MYAPP

\* class math\_stuff

\*/

MYAPP.math\_stuff = {

/\*\*

\* Sums two numbers

\*

\* @method sum

\* param {Number} 是第一个数

\* param {Number} 是第二个数

\* return {Number} 两个输入的总和

\*/

sum: function (a, b) {

return a + b;

},

/\*\*

\* Multiplies two numbers

\* param {Number} 是第一个数

\* param {Number} 是第二个数

\* return {Number} 两个输入相乘后结果

\*/

multi: function (a, b) {

return a \* b;

}

};

@namespace：这里用于命名包含以上对象的全局引用的名称

@class：这里有些命名不当，他实际意思是指对象或者构造函数

@method：定义对象中的方法和方法名

@param：列举函数所使用的参数。其中将参数类型用大括号括起来，并在其后注释参数名及描述。

@return：类似于@param，这里用于描述返回值的，并且该方法没有名称。

@constructor：表明这个“类”实际上是一个构造函数

@property和@type描述了对象的属性。

### 2. 编写API目的：

* 为API编写注释不仅仅是一中提供参考文档的简便方法，而且还有其他用途——通过再次审查代码，提高代码质量。
* 在解决问题时写出的解决方案仅仅是一个初稿。该解决方案可以给出令人期待的输出，但是该方案是否是最佳方案呢？改代码是否可读、易于理解、维护和升级呢？当您再次审视代码时您将更加确定代码哪些部分可以改进——如何使得代码更容易继续更新，移除一些不足之处等。它可以极大地帮助您创建高质量的代码。

data = this.json.data; // 缓存作用域链较深的对象

}

//当需要缓存this时必须使用self变量进行缓存

// 缓存this

function Row(name) {

var self = this;

self.name = name;

$(".row").click(function() {

self.getName();

});

}

self是一个保留字，不过用它也没关系。在这里，看个人爱好吧，可以用\_this, that, me等这些词，都行，但是团队开发的时候统一下比较好。

使用eval，采取$.parseJSON

三个原因：

* + 有注入风险，尤其是ajax返回数据
  + 不方便debug
  + 效率低，eval是一个执行效率很低的函数

建议： 使用new Function来代替eval的使用，最好就别用。

## 六、需要注意的地方

window.onload只能使用一次，使用多次会被最后的覆盖。

解决方案：

* + 只引用一次window.onload
  + 使用jQuery的$(docuemnt.ready(function(){}); 可多次使用
  + 使用函数封装，代码地址

CSS放在页头引入，javascript放在页尾引入

1. 在上线之前，要编译压缩代码
2. 减少重排与重绘