**编程注释规范**

黄金定律——不管有多少人共同参与同一项目，一定要确保每一行代码都像是同一个人编写的。

**注释规范**

编码规范中有一条很重要---见名知意，有时候直白的函数名还不能表达完整的意思，这个时候注释就出场了，代码是由人编写并维护的。请确保你的代码能够自描述、注释良好并且易于他人理解。好的代码注释能够传达上下文关系和代码目的。

在写代码前就应该添加注释，这时在你的脑子里的是清晰完整的思路。

如果在代码最后再添加注释，它将花费你双倍的时间。

**注释的写法**

html

<!-- 注释的内容 -->

css

/\* declarations \*/

javascript、less

// 注释内容

/\* 注释内容 \*/

**通用**

注释尽量使用英文来描述

**单行注释**

·双斜线后，必须跟注释内容保留一个空格

·可独占一行, 前边不许有空行（还是可以有？）, 缩进与下一行代码保持一致

·可位于一个代码行的末尾，注意这里的格式

// Good

if (condition) {

// if you made it here, then all security checks passed

allowed();

}

var zhangsan = "zhangsan"; // 双斜线距离分号四个空格，双斜线后始终保留一个空格

**多行注释格式**

·最少三行, 格式如下

·前边留空一行

**何时使用**

·难于理解的代码段

·可能存在错误的代码段

·浏览器特殊的HACK代码

·想吐槽的产品逻辑, 合作同事

·业务逻辑强相关的代码

/\*

\* 注释内容与星标前保留一个空格

\*/

**文件注释**

/\*

\* @author: who are you

\* @date: when you write it

\* @description: the function of this file.

\*/

**多人合作注释**

同一个文件的代码可能被多个人修改，这个时候需要标识出谁改动了哪些部分。

格式： // add begin by 作者名 ,一个分号;，再加上原因 Reason

代码添加的最后加上：//add end

Example

// add begin by liuxing ; Init post's id

var postId = 1;

//end add

或者

// add begin by liuxing

/\*\*

\* 多行注释来说明原因

\*/

var postId = 1;

//end add

**javascript 注释**

**文档注释**

·各类标签 @param @method 等 参考 http://usejsdoc.org/

·格式如下

**用在哪里**

·所有的方法

·所有的构造函数

·所有的全局变量

/\*\*

\* here boy, look here , here is girl

\* @method lookGril

\* @param {Object} balabalabala

\* @return {Object} balabalabala

\*/

·常用tag

@author、@callback、@copyright、@default、@deprecated、@throws 、@todo

@param、@returns

**方法与函数的注释**

提供参数的说明. 使用完整的句子, 并用第三人称来书写方法说明.

/\*\*

\* Converts text to some completely different text.

\* @param {string} arg1 An argument that makes this more interesting.

\* @return {string} Some return value.

\*/

project.MyClass.prototype.someMethod = function(arg1) {

// ...

};

**属性注释**

也需要对属性进行注释.

/\*\*

\* Maximum number of things per pane.

\* @type {number}

\*/

project.MyClass.prototype.someProperty = 4;

**枚举**

/\*\*

\* Enum for tri-state values.

\* @enum {number}

\*/

project.TriState = {

TRUE: 1,

FALSE: -1,

MAYBE: 0

};

注意一下, 枚举也具有有效类型, 所以可以当成参数类型来用.

/\*\*

\* Sets project state.

\* @param {project.TriState} state New project state.

\*/

project.setState = function(state) {

// ...

};

**Typedefs**

有时类型会很复杂. 比如下面的函数, 接收 Element 参数:

/\*\*

\* @param {string} tagName

\* @param {(string|Element|Text|Array.<Element>|Array.<Text>)} contents

\* @return {Element}

\*/

goog.createElement = function(tagName, contents) {

...

};

**HTML注释**

**CSS 注释**