# HTML5开发规范文档2.0

[1 目的 3](#_Toc400802619)

[2 适用范围 4](#_Toc400802620)

[3 命名 4](#_Toc400802621)

[3.1命名法 4](#_Toc400802622)

[3.2 命名规则 4](#_Toc400802623)

[3.2.1 文件命名 4](#_Toc400802624)

[3.2.2 变量与参数命名 4](#_Toc400802625)

[3.2.3 方法 5](#_Toc400802626)

[3.3 关键字 5](#_Toc400802627)

[3.3.1 php关键字 5](#_Toc400802628)

[3.3.2 javascript关键字 6](#_Toc400802629)

[4 html5结构 7](#_Toc400802630)

[4.1 文件分层 7](#_Toc400802631)

[4.1.1.php层。 7](#_Toc400802632)

[4.1.2.html层。 8](#_Toc400802633)

[4.1.3.public公共区域。 8](#_Toc400802634)

[4.1.4．tools 8](#_Toc400802635)

[4.2 流程介绍 8](#_Toc400802636)

[5.注释以及日志 8](#_Toc400802637)

[6 代码排版 13](#_Toc400802638)

[6.1 间隔与格式化 13](#_Toc400802639)

[6.2 行长度 13](#_Toc400802640)

[6.3花括号 13](#_Toc400802641)

[6.4 空格与空行 13](#_Toc400802642)

[7 修改规范 14](#_Toc400802643)

[7.1修改代码 14](#_Toc400802644)

[7.2新增代码行 14](#_Toc400802645)

[7.3 删除代码行 15](#_Toc400802646)

# 1 目的

统一编程风格，提高的可读性与编码效率，提高软件源程序的质量和可维护性，避免团队开发可能带来的混乱，减少软件维护成本。

# 2 适用范围

适用所有HTML5项目。

# 3 命名

## 3.1命名法

采用英文单词或其组合，便于记忆和阅读，切忌使用汉语拼音来命名。使用完整单词，避免缩写词（除非该缩写词被更广泛使用，像 URL，HTML）。

1.驼峰命名法：

就是当变量名或函式名是由一个或多个单字连结在一起，而构成的唯一识别字时，驼峰命名法第一个单字以小写字母开始；第二个单字的首字母大写或每一个单字的首字母都采用大写字母

2.下划线命名法：

与驼峰命名法相比，将每个大写字母改为下划线形式。

## 3.2 命名规则

### 3.2.1 文件命名

1）Php文件一律小写，以实现的功能的英文作为文件名。

2）html，css和js文件，可以按照上述命名法。

3）由美工提交过来的html静态文件（public下），可以用中文标识。注意：以功能名称 + 下划线 + 日期 +下划线 + 时间 + （特殊说明）。如:

1.机票列表\_2014-10-10\_1120.html

2.机票列表\_2014-10-10\_1120 (加急).html

### 3.2.2 变量与参数命名

变量用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。变量名应简短且富于描述。变量名的选用应该易于记忆，即：能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。如for循环里面的i，j。

1.Php变量以$符号开头，不需要声明类型 $str = “hello”;

2.javascript变量遵循java变量声明规则 var str = “hello”;

3.jquery的$符号为全局变量有其特殊含义，不可混淆。$(‘body’).css(‘background-color’,’red’);

### 3.2.3 方法

1.名称要使用意义完整的英文描述：可以使用驼峰命名法或者下划线命名法。如：

public function orderList(){}

public function order\_list(){}

注意，在一个类名下，方法命名要统一，不得出现下划线和驼峰命名法混淆的现象。

2.对于存取属性的方法，采用set和 get方法，动作方法采用动词和动宾结构，如：

get + 非布尔属性名()

is + 布尔属性名()

set + 属性名()

动词()

动词 + 宾语()

## 3.3 关键字

### 3.3.1 php关键字

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [and](http://php.net/manual/zh/language.operators.logical.php) | [or](http://php.net/manual/zh/language.operators.logical.php) | [xor](http://php.net/manual/zh/language.operators.logical.php) | [\_\_FILE\_\_](http://php.net/manual/zh/language.constants.predefined.php) | [exception](http://php.net/manual/zh/reserved.classes.php#reserved.classes.php5) (PHP 5) |
| [\_\_LINE\_\_](http://php.net/manual/zh/language.constants.predefined.php) | [array()](http://php.net/manual/zh/function.array.php) | [as](http://php.net/manual/zh/control-structures.foreach.php) | [break](http://php.net/manual/zh/control-structures.break.php) | [case](http://php.net/manual/zh/control-structures.switch.php) |
| [class](http://php.net/manual/zh/keyword.class.php) | [const](http://php.net/manual/zh/language.oop5.constants.php) | [continue](http://php.net/manual/zh/control-structures.continue.php) | [declare](http://php.net/manual/zh/control-structures.declare.php) | [default](http://php.net/manual/zh/control-structures.switch.php) |
| [die()](http://php.net/manual/zh/function.die.php) | [do](http://php.net/manual/zh/control-structures.do.while.php) | [echo](http://php.net/manual/zh/function.echo.php) | [else](http://php.net/manual/zh/control-structures.else.php) | [elseif](http://php.net/manual/zh/control-structures.elseif.php) |
| [empty()](http://php.net/manual/zh/function.empty.php) | [enddeclare](http://php.net/manual/zh/control-structures.declare.php) | [endfor](http://php.net/manual/zh/control-structures.alternative-syntax.php) | [endforeach](http://php.net/manual/zh/control-structures.alternative-syntax.php) | [endif](http://php.net/manual/zh/control-structures.alternative-syntax.php) |
| [endswitch](http://php.net/manual/zh/control-structures.alternative-syntax.php) | [endwhile](http://php.net/manual/zh/control-structures.alternative-syntax.php) | [eval()](http://php.net/manual/zh/function.eval.php) | [exit()](http://php.net/manual/zh/function.exit.php) | [extends](http://php.net/manual/zh/keyword.class.php) |
| [for](http://php.net/manual/zh/control-structures.for.php) | [foreach](http://php.net/manual/zh/control-structures.foreach.php) | [function](http://php.net/manual/zh/functions.user-defined.php) | [global](http://php.net/manual/zh/language.variables.scope.php) | [if](http://php.net/manual/zh/control-structures.if.php) |
| [include](http://php.net/manual/zh/function.include.php) | [include\_once](http://php.net/manual/zh/function.include-once.php) | [isset()](http://php.net/manual/zh/function.isset.php) | [list()](http://php.net/manual/zh/function.list.php) | [new](http://php.net/manual/zh/keyword.class.php) |
| [print](http://php.net/manual/zh/function.print.php) | [require](http://php.net/manual/zh/function.require.php) | [require\_once](http://php.net/manual/zh/function.require-once.php) | [return](http://php.net/manual/zh/function.return.php) | [static](http://php.net/manual/zh/language.variables.scope.php) |
| [switch](http://php.net/manual/zh/control-structures.switch.php) | [unset()](http://php.net/manual/zh/function.unset.php) | use | [var](http://php.net/manual/zh/keyword.class.php) | [while](http://php.net/manual/zh/control-structures.while.php) |
| [\_\_FUNCTION\_\_](http://php.net/manual/zh/language.constants.predefined.php) | [\_\_CLASS\_\_](http://php.net/manual/zh/language.constants.predefined.php) | [\_\_METHOD\_\_](http://php.net/manual/zh/language.constants.predefined.php) | [final](http://php.net/manual/zh/language.oop5.final.php) (PHP 5) | [php\_user\_filter](http://php.net/manual/zh/reserved.classes.php#reserved.classes.php5) (PHP 5) |
| [interface](http://php.net/manual/zh/language.oop5.interfaces.php) (PHP 5) | [implements](http://php.net/manual/zh/language.oop5.interfaces.php) (PHP 5) | [extends](http://php.net/manual/zh/language.oop5.basic.php) | [public](http://php.net/manual/zh/language.oop5.visibility.php) (PHP 5) | [private](http://php.net/manual/zh/language.oop5.visibility.php) (PHP 5) |
| [protected](http://php.net/manual/zh/language.oop5.visibility.php) (PHP 5) | [abstract](http://php.net/manual/zh/language.oop5.abstract.php) (PHP 5) | [clone](http://php.net/manual/zh/language.oop5.cloning.php) (PHP 5) | [try](http://php.net/manual/zh/language.exceptions.php) (PHP 5) | [catch](http://php.net/manual/zh/language.exceptions.php) (PHP 5) |
| [throw](http://php.net/manual/zh/language.exceptions.php) (PHP 5) | cfunction (PHP 4 only) | [this](http://php.net/manual/zh/keyword.class.php) (PHP 5 only) |  |  |

### 3.3.2 javascript关键字

1.关键字：

break  
case  
catch  
continue  
default  
delete  
do  
else  
finally  
for  
function  
if  
in  
instanceof  
new  
return  
switch  
this  
throw  
try  
typeof  
var  
void  
while  
with

2.保留字  
abstract  
boolean  
byte  
char  
class  
const  
debugger  
double  
enum  
export  
extends  
final  
float  
goto  
implements  
import  
int  
interface  
long  
native  
package  
[pr](http://pr.knowsky.com/)ivate  
protected  
public  
short  
static  
super  
synchronized  
throws  
transient  
volatile

# 4 html5结构

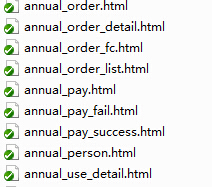
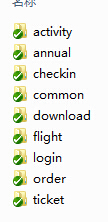
## 4.1 文件分层

### 4.1.1.php层。

以开源框架ci为基础，做数据交互层。数据返回统一为json或xml。

### 4.1.2.html层。

存放经过渲染静态文件，按照模块存放。



### 4.1.3.public公共区域。

主要包括css，js和image文件。其中，在特殊划定的html文件夹下面，是美工写入的带有中文名的静态文件。

### 4.1.4．tools

独立性比较强的js控件，如日历控件，scroll控件等，整理好，归档在此文件夹下，以便重复调用。

## 4.2 流程介绍

1.由html层的静态网页作为入口。

2.通过ajax方式与后台交互。

3.拿到json数据后，解析，并渲染文件。

# 5.注释以及日志

注释有助于理解代码，有效的注释是指在代码的功能、意图层次上进行注释，提供有用、额外的信息，而不是代码表面意义的简单重复。

实现注释有4种实现注释的风格：

* 块注释(BlockComments)

块注释被置于每个文件的开始处以及每个方法之前。它们也可以被用于其他地方，比如方法的内部。在功能和方法内部的块注释应该和它们所描述的代码具有一样的缩进格式。

* 单行注释(single-line Comments)

短注释可以显示一行内，并与其后的代码具有一样的缩进层级。如果一个注释不能在一行内写完，就该块注释(参见“块注释”)。单行注释之前应该有一个空行。以下是一个Java代码中单行注释的例子：

if (condition) {

/ \* special case \*/

……

}

* 尾端(trailing)

极短的注释可以与它们所要描述的代码位于同一行，但是应该有足够的空白来分开代码和注释。以下是一个js代码中单行注释的例子：

if (a ==2) {

return true; / \* special case \*/

}

* 行末(end-of-line)。

注释界定符“//”，可以注释掉整行或者一行中的一部分。它一般不用于连续多行的注释文本；然而，它可以用来注释掉多行的代码段。

1. 块注释之首应该有一个空行，用于把块注释和代码分割开来

/\*\*

\*注释

\*/

/\*避免这样的注释

\*/

1. 注释符与注释内容之间要用一个空格进行分隔。

/\*注释内容 \*/

// 注释内容

/\*避免这样的注释\*/

//注释内容

1. 注释应与其描述的代码相近，对代码的注释应放在其上方（需与其上面的代码用空行隔开）或右方（对单条语句的注释）相邻位置，不可放在下面。

在使用缩写时或之前，应对缩写进行必要的说明。避免在代码行的末尾添加注释；行尾注释使代码更难阅读。不过在批注变量声明时，行尾注释是合适的；在这种情况下，将所有行尾注释在公共制表位处对齐。

// 注释内容

Var str = “abc”;

// 代码段1注释

[ 代码段1 ]

/\* 代码段2注释 \*/

[ 代码段2 ]

如下例子注释与描述的代码相隔太远。

/\* 注释内容 \*/

Var str = “abc”;

如下例子注释不应放在所描述的代码下面。

Var str = “abc”;

/\* 注释内容 \*/

如下例子，显得代码与注释过于紧凑。

/\* 代码段1注释 \*/

[ 代码段1 ]

/\* 代码段2注释 \*/

[ 代码段2 ]

1. 单行注释与所描述内容进行同样的缩进。

Pubic function doSomething(){

/\* 代码段1注释 \*/

[ 代码段1 ]

/\* 代码段2注释 \*/

[ 代码段2 ]

}

pubic function doSomething() {

/\* 代码段1注释 \*/

[ 代码段1 ]

/\* 代码段2注释 \*/

[ 代码段2 ]

}

1. 若有多个尾端注释出现于大段代码中，它们应该具有相同的缩进。

if (condition) {

[ 代码段1 ]/\* 代码段1注释 \*/

}else {

[ 比较长的代码段2 ]/\* 代码段2注释 \*/

}

1. 包含在{ }中代码块的结束处要加注释，便于阅读。特别是多分支、多重嵌套的条件语句或循环语句。

**说明：**此时注释可以用英文，方便查找对应的语句。

public function doSomething {

if (condition1) {

for (……) {

if (condition2) {

} // end of if (condition2)

} // end of for (……)

} // end of if (condition1)

} // enddoSomething

1. 对分支语句（条件分支、循环语句等）必须编写注释。

**说明：**这些语句往往是程序实现某一特殊功能的关键，对于维护人员来说，良好的注释有助于更好的理解程序，有时甚至优于看设计文档。

1. 单元文件的文件头注释说明应按如下格式：

/\*\*

\* 文件名

\* Author: 张三

\* Date: 2014-2-27 上午11:43:11

\* Copyright (c) 2014年 佰邦达科技(北京)有限公司. All rights reserved.

\* 功能描述：功能描述

\* 修改记录：(仅记录功能修改)

\* 小明 2014-03-01 添加某某属性

\*/

1. 不必每行都加注释，在3~10行左右的段落做注释要好于每行都做注释，显而易见的代码不加注释 。

# 6 代码排版

## 6.1 间隔与格式化

可以使用Tab键进行缩排。

## 6.2 行长度

代码中的每行文本不要超过80个字符长度。

## 6.3花括号

风格与项目组保持一致，if条件后面即使只有一行代码，也加上花括弧

//建议用法

for (var i = 0; i < 10; i++){

……

}

## 6.4 空格与空行

1.关键字与其后的表达式之间要有空格，如：

if (expr)

或

for (expr)

1. 一元操作符如“++”、“--”、“!”、“~”、（类型）等前后不加空格。“[]”“.”这类操作符前后不加空格。
2. 多元运算符和它们的操作数之间至少需要一个空格。

例如

i=i+1; //错误的写法，操作符两端没有空格

i = i + 1; //正确的写法，

if(a>b) //错误的写法，逻辑判断符号两端没有空格

if(a > b) //正确的写法

1. 不同逻辑程序块之间要使用空行分隔。

空行起着分隔程序段落的作用。适当的空行可以使程序的布局更加清晰。

1. 不同逻辑程序块之间要使用空行分隔。

空行起着分隔程序段落的作用。适当的空行可以使程序的布局更加清晰。

下列情况应该总是使用空行：

1. 一个源文件的两个片段(section)之间
2. 类声明和接口声明之间
3. 常量声明区域之后
4. 方法声明之前
5. 方法内的局部变量和方法的第一条语句之间
6. 一个方法内的两个逻辑段之间，用以提高可读性

# 7 修改规范

## 7.1修改代码

修改代码行以注释旧代码行后再新增代码行的方式进行。

//修改人，修改时间，修改说明

//修改前代码行开始

//修改前代码行

//修改前代码行结束

//修改后代码行开始

修改后代码行

## 7.2新增代码行

新增代码行的前后应有注释行说明。

//修改人，修改时间，修改说明

新增代码行

//修改结束

7.3 删除代码行

删除代码向的前后用注释行说明

//修改人，修改时间，修改说明

要删除的代码行(将要删除的语句进行注释)

//修改结束

# 8 html + css排版要求

8.1 基本规范：

* 用两个空格来代替制表符（tab） -- 这是唯一能保证在所有环境下获得一致展现的方法。
* 嵌套元素应当缩进一次（即两个空格）。
* 对于属性的定义，确保全部使用双引号，绝不要使用单引号。
* 不要在自闭合（self-closing）元素的尾部添加斜线 --html5规范中明确说明这是可选的。
* 不要省略可选的结束标签（closing tag）（例如，</li> 或 </body>）。

8.2 语言属性

根据 HTML5 规范：

强烈建议为 html 根元素指定 lang 属性，从而为文档设置正确的语言。这将有助于语音合成工具确定其所应该采用的发音，有助于翻译工具确定其翻译时所应遵守的规则等等。

IE 支持通过特定的 <meta> 标签来确定绘制当前页面所应该采用的 IE 版本。除非有强烈的特殊需求，否则最好是设置为 **edge mode**，从而通知 IE 采用其所支持的最新的模式。

<meta http-equiv="X-UA-Compatible" content="IE=Edge">

**8.3 字符编码**

通过明确声明字符编码，能够确保浏览器快速并容易的判断页面内容的渲染方式。这样做的好处是，可以避免在 HTML 中使用字符实体标记（character entity），从而全部与文档编码一致（一般采用 UTF-8 编码）。

<head> <meta charset="UTF-8"> </head>

**8.4 引入 CSS 和 JavaScript 文件**

根据 HTML5 规范，在引入 CSS 和 JavaScript 文件时一般不需要指定 type 属性，因为 text/css 和 text/javascript 分别是它们的默认值。 <link rel="stylesheet" href="code-guide.css"> <!-- In-document CSS --> <style> /\* ... \*/ </style> <!-- JavaScript --> <script src="code-guide.js"></script>

**8.5 实用为王**

尽量遵循 HTML 标准和语义，但是不要以牺牲实用性为代价。任何时候都要尽量使用最少的标签并保持最小的复杂度。

**8.6属性顺序**

HTML 属性应当按照以下给出的顺序依次排列，确保代码的易读性。

* class
* id, name
* data-\*
* src, for, type, href
* title, alt
* aria-\*, role

class 用于标识高度可复用组件，因此应该排在首位。id 用于标识具体组件，应当谨慎使用（例如，页面内的书签），因此排在第二位。

<a class="..." id="..." data-modal="toggle" href="#"> Example link </a> <input class="form-control" type="text"> <img src="..." alt="...">

## 8.7 布尔（boolean）型属性

布尔型属性可以在声明时不赋值。XHTML 规范要求为其赋值，但是 HTML5 规范不需要。

元素的布尔型属性如果有值，就是 true，如果没有值，就是 false。

如果一定要为其赋值的话，请参考 WhatWG 规范：

如果属性存在，其值必须是空字符串或 [...] 属性的规范名称，并且不要再收尾添加空白符。

**简单来说，就是不用赋值。**

<input type="text" disabled> <input type="checkbox" value="1" checked> <select> <option value="1" selected>1</option> </select>

## 8.8 减少标签的数量

编写 HTML 代码时，尽量避免多余的父元素。很多时候，这需要迭代和重构来实现。请看下面的案例：

<!-- Not so great --> <span class="avatar"> <img src="..."> </span> <!-- Better --> <img class="avatar" src="...">

## 8.9 JavaScript 生成的标签

通过 JavaScript 生成的标签让内容变得不易查找、编辑，并且降低性能。能避免时尽量避免。