
FRONT END ST
Web前端开发
e W3C community provides a great service through volun
eases, and other documentation. You can search the tran
nslation (or collaborate with someone else wishing to tran
o help keep W3C an international organization.

序

✍Standards for developing flexible, durable, and sustainable HTML and CSS, and maintainable JavaScript（开发灵活的、持久的和可持续的HTML和CSS,和可维护的JavaScript的标准！）————Code Guide by @AlloyTeam

✍ 任何程序员都有自己的开发习惯

后来想想任何人（甚至小动物）都有自己的做某件事情的习惯，而对于开发而言，所谓的标准规范，其实就是程序员对自身的要求，和对团队的融入，效率，质量是每个团队的追求。

简述：要自己写的代码，别人容易看懂！　　如果使用其他系统或其他人早已定义好的变量、类、函数名时，需遵循它们原本的形式，避免出现问题。

✍ 摘自（阮一峰）

所谓"编程风格"（programming style），指的是编写代码的样式规则。不同的程序员，往往有不同的编程风格。

有人说，编译器的规范叫做"语法规则"（grammar），这是程序员必须遵守的；而编译器忽略的部分，就叫"编程风格"（programming style），这是程序员可以自由选择的。这种说法不完全正确，程序员固然可以自由选择编程风格，但是好的编程风格有助于写出质量更高、错误更少、更易于维护的程序。

所以，有一点应该明确，*"编程风格"的选择不应该基于个人爱好、熟悉程度、打字工作量等因素，而要考虑如何尽量使代码清晰易读、减少出错*。你选择的，不是你喜欢的风格，而是一种能够清晰表达你的意图的风格。

✍ 规范目的

为提高团队协作效率, 便于后台人员添加功能及前端后期优化维护, 输出高质量的文档, 特制订此文档。本规范文档一经确认, 前端开发人员必须按本文档规范进行前台页面开发。本文档如有不对或者不合适的地方请及时提出, 经讨论决定后方可更改。

> 1、程序员可以了解任何代码，弄清程序的状况；
> 2、新人可以很快的适应环境；
> 3、防止新接触PHP的人出于节省时间的需要，自创一套风格并养成终生的习惯；
> 4、防止新接触PHP的人一次次的犯同样的错误；
> 5、在一致的环境下，人们可以减少犯错的机会；
> 6、程序员们有了一致的敌人；

✍ 基本准则

- 保证开发的有效性和合理性，并可最大程度的提高程序代码的可读性和可重复利用性。
- 符合web标准, 语义化html, 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。

开发注意点

> 要求以UTF-8无BOM信息头格式保存html文件、css文件、js文件、php文件。(否则可能导致很多意想不

- 如果使用!important使一个属性生效，那么可以确定发生了层叠优先级问题。
- 因为xhtml区分大小写，可以总是选择使用小写值，以避免出现意外错误。
- 如果不确定一个样式表是否被加载，那么可以在样式表中添加一条特殊规则，然后再检查它是否可以成功应用。这条规则通常要设置非常显眼的效果，如*{border:1px solid black;}。
- PHP中:在定义字符串变量时,应尽量使用单引号.单/双引号包含的HTML标签,输出结果一样.两者的不同之处在于,双引号中所包含的变量会自动被替换成实际数值,而在单引号中包含的变量则按普通字符串输出。
- JavaScript中使用'',因为W3C规定HTML属性使用"",避免两者混淆,CSS文件使用"";
- 反引号(后引号)(`):位于大多数键盘左上角的波浪号之下;slash:斜线[杠] (/);
- 单引号（ ' ）、双引号（ " ）、反斜线（ \ ）与 NUL（NULL 字符）;
  同时要注意的是一个 NULL 字节（ "\0" ）并不等同于 PHP 的 NULL 常数;
- 动态HTML,或者DHTML,是一个用来描述将静态HTML,层叠样式表(CSS)以及JavaScript结合,并且在载入一个静态Web页面所有元素后,通过文档对象模型修改页面外观的术语;
- 对于PHP,java的命名规则:类:每个单词首字母大写,方法:第一个单词首字母小写,其他首字母大写;JavaScript要求不严,但最好实际编写时遵循以上规则。

★三者共同点（不要求，但是希望读者，养成好习惯！）：
在定义类时，全部单词的首字母必须大写；例如：Person，ClassDemo;（帕斯卡命名法）
在定义函数(方法)时也有命名规范要求，即第一个单词的首字母小写，之后每个单词的首字母大写，如printInfo()方法。（驼峰命名法）
每个函数都有唯一的名称；

◆Java中的变量名的标识符:不能以数字开头，可以由英文字母、数字、下划线（ _ ）和美元符号($)组成，不能是保留关键字，区分大小写；

◆PHP中的变量名的标识符:不能以数字开头，可以由英文字母、数字和下划线（ _ ）组成，不能是保留关键字，区分大小写；但是内置结构和关键字以及用户自定义的类名和函数名都是不区分大小写的。

◆JavaScript 变量名称的规则：
●变量对大小写敏感，区分大小写【y 和 Y 是两个不同的变量】
●变量必须以字母或下划线开始。

注意：无参数的函数必须在其函数名后加括号：

```
function 函数名()
 {
   代码．．．
 }
```

注意：别忘记 JavaScript 中大小写字母的重要性。"function" 这个词必须是小写的，否则 JavaScript 就会出错。另外需要注意的是，必须使用大小写完全相同的函数名来调用函数。

ⅢⅢ
ⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢⅢ

■程序中关于命名主要有三种方法：骆驼命名法、帕斯卡命名法（有人称之为“大驼峰式命名法”Upper Camel Case）、匈牙利命名法；

【骆驼命名法，驼峰法】 就是第一个字母要小写，后面的单词的第一个字母就要用大写，如下：
navMenuRedButton

【帕斯卡命名法】
所有单词的首字母都要大写，如下：
NavMenuRedButton

【匈牙利命名法】
在名称前面加上一个或多个小写字母作为前缀，来让名称更加好认，更容易理解，比如：
head_navigation
red_navMenuButton

■以上三种，前两种（骆驼命名法、帕斯卡命名法）在命名的时候比较常用，当然这三种命名法可以混合使用。

文件、目录结构规范

文件夹及文件名命名

建议在开发规范的独立的PHP项目时，使用规范的文件目录结构，这有助于提高项目的逻辑结构合理性，对应扩展和合作，以及团队开发均有好处。

程序文件名和目录名命名均采用有意义的英文方式命名，不使用拼音或无意义的字母，同时均必须使用小写字母，多个词间使用_间隔。

系统结构：

```
[PHPWEB]  // 系统根目录
    |--Api          // 接口文件目录
    |--Apps         // 应用模块目录
    |--Core         // 核心框架目录（建议将框架放置在网站目录外，安全）
    |--Doc          // 项目相关文档目录
    |--Data         // 数据文件存放目录
    |--Runtime      // 系统运行时文件目录
    |--Statics      // (或者Public)静态资源包
        |--css          // css文件存放目录
            |--img          // css中用到的图片文件存放目录
        |--images       // 所有图片文件存放路径（在里面根据目录结构设立子目录）
        |--js           // js脚本存放目录
    |--theme        // 主题目录
        |--default      // 默认主题目录
        |--...          // 其他主题目录
    |--Uploads      // 上传文件目录
    |--crossdomain.xml      // FLASH跨域传输文件
    |--robots.txt           // 搜索引擎蜘蛛限制配置文件
    |--favicon.ico          // 系统icon图标
```

以上目录结构是通常的目录结构，根据具体应用的具体情况，可以考虑不用完全遵循，但是尽量做到规范化。

> 不需要直接暴露给用户的文件，应该放在Web服务器访问不到的目录，避免因为配置问题而泄露设置信息。

安全规范

安全规范

当我们尝试编码时，很多时候不知道如何去让自己的代码变得安全一点，因为我们缺乏安全常识，安全常识的规范可以帮你杜绝一些日常的菜鸟黑客的攻击，却不能阻止骨灰级专家们的凌厉攻势，所以更高深的安全我们还得从其他途径学习。

安全规则

输入和输出

检查是否做了HTML代码的过滤 可能出现的问题：如果有人输入恶意的HTML代码，会导致窃取 `cookie`，产生恶意登录表单，和破坏网站。

检查变量做数据库操作之前是否做了 `escape` 可能出现的问题：如果一个要写入查询语句的字符串变量包含了某些特殊的字符，比如引号(',")或者分号(;) 可能造成执行了预期之外的操作。
建议采用的方法：使用 `mysql_escape_string()` 或实现类似功能的函数。

检查输入数值的合法性 可能出现的问题：异常的数值会造成问题。如果对输入的数值不做检查会造成不合法的或者错误的数据存入UDB、存入其它的数据库或者导致意料之外的程序操作发生。
举例：如果程序以用户输入的参数值做为文件名，进行文件操作，恶意输入系统文件名会造成系统损毁。

核实对cookie的使用以及对用户数据的处理 可能出现的问题：不正确的 `cookie` 使用可能造成用户数据泄漏。

先申明后使用

XXX环境下的 PHP 代码编写要求所有的变量均需要先申明后使用，否则会有错误信息，对于数组，在使用一个不确定的 `key` 时，比如先进行 `isset()` 的判断，然后再使用；比如下面的代码：

```
$array = array();
$var = isset( $array[3] ) ? $array[3] : '' ;
```

严格的过滤和合法性验证

在XXX环境下，对 web 通过 GET 或者 POST 方法传递来的参数均要求进行 严格的过滤和合法性验证，不推荐使用直接的 $_GET 、 $_POST 或者 $_REQUEST 获取，而通过 XXX 的 XXX_yiv 模块提供的方法获

取和过滤处理，（类似ThinkPHP的I函数）。

访问控制

对内部使用的产品或者供合作方使用的产品，要考虑增加访问控制。

logs

确保用户的保密信息没有记在log中(例如：用户的密码)；
确保对关键的用户操作保存了完整的用户访问记录。

https

对敏感数据的传输要采用https。

针对php的规则

设置 `register_globals = off`；
设置 `error_reporting = E_ALL` ，并且要修正所有的 `error` 和 `warning`；
将实际的操作放在被引用的文件中。把引用文件放到不可以被直接浏览的目录下。

> register_globals已自 PHP 5.3.0 起废弃并将自 PHP 5.4.0 起移除。

当 `register_globals=Off` 的时候，下一个程序接收的时候应该用 `$_GET['user_name']` 和 `$_GET['user_pass']` 来接受传递过来的值。（注：当 `<form>` 的method属性为post的时候应该用 `$_POST['user_name']` 和 `$_POST['user_pass']` ）

当 `register_globals=On` 的时候，下一个程序可以直接使用 `$user_name` 和 `$user_pass` 来接受值。

顾名思义，register_globals的意思就是注册为全局变量，所以当On的时候，传递过来的值会被直接的注册为全局变量直接使用，而Off的时候，我们需要到特定的数组里去得到它。所以，碰到上边那些无法得到值的问题的朋友应该首先检查一下你的register_globals的设置和你获取值的方法是否匹配。（查看可以用phpinfo()函数或者直接查看php.ini）。

HTML开发规范

> 参考：华南城网前端编码规范 1.0

xhtml区分大小写，xhtml要求标签名、属性名、值都要小写，并且要有双引号和标签闭合。
因为XHTML文档是XML应用产物，XML是区分大小写的，所以 `<br>` 和 `<BR>` 会被认为是两种不同的标签。

> 注意：HTML5不区分大小写！

1、DOCTYPE

页面文档类型统一使用HTML5 DOCTYPE. 代码如下：

```
<!DOCTYPE html>
```

2、编码

声明方法遵规循HTML5的范. 代码如下：

```
<meta charset="utf-8" />
```

3、语义化

使用符合语义的标签书写 HTML 文档, 选择恰当的元素表达所需的含义。

- 结构性元素
  - `p` 表示段落. 只能包含内联元素, 不能包含块级元素;
  - `li` 本身无特殊含义, 可用于布局. 几乎可以包含任何元素;
  - `br` 表示换行符;
  - `hr` 表示水平分割线;
  - `h1` - `h6` 表示标题. 其中 h1 用于表示当前页面最重要的内容的标题;
  - `blockquote` 表示引用, 可以包含多个段落. 请勿纯粹为了缩进而使用blockquote, 大部分浏览器

默认将 blockquote 渲染为带有左右缩进;
    - `pre` 表示一段格式化好的文本；

- 头部元素
    - `title` 每个页面必须有且仅有一个 title 元素;
    - `base` 可用场景：首页、频道等大部分链接都为新窗口打开的页面;
    - `link` 用于引入 css 资源时, 可省去 media(默认为all) 和 type(默认为text/css) 属性;
    - `style` `type` 默认为 text/css, 可以省去;
    - `script` `type` 属性可以省去; 不赞成使用lang属性; 不要使用古老的这种hack脚本, 它用于阻止第一代浏览器(Netscape 1和Mosaic)将脚本显示成文字;
    - `noscript` 在用户代理不支持 JavaScript 的情况下提供说明；

- 文本元素
    - `a` 存在 href 属性时表示链接, 无 href 属性但有 name 属性表示锚点;
    - `em,strong` em 表示句意强调, 加与不加会引起语义变化, 可用于表示不同的心情或语调; strong 表示重要性强调, 可用于局部或全局, strong强调的是重要性, 不会改变句意;
    - `abbr` 表示缩写;
    - `sub,sup` 主要用于数学和化学公式, sup还可用于脚注;
    - `span` 本身无特殊含义;
    - `ins,del` 分别表示从文档中增加(插入)和删除；

- 媒体元素
    - `img` 请勿将img元素作为定位布局的工具, 不要用他显示空白图片; 必要时给img元素增加alt属性;
    - `object` 可以用来插入Flash;

- 列表元素
    - `dl` 表示关联列表, dd是对dt的解释; dt和dd的对应关系比较随意：一个dt对应多个dd、多个dt对应一个dd、多个dt对应多个dd, 都合法; 可用于名词/单词解释、日程列表、站点目录;
    - `ul` 表示无序列表;
    - `ol` 表示有序列表, 可用于排行榜等;
    - `li` 表示列表项, 必须是ul/ol的子元素;

- 表单元素
    - 推荐使用 `button` 代替 input, 但必须声明 type;
    - 推荐使用 `fieldset`, `legend` 组织表单；
    - 表单元素的 `name` 不能设定为 action, enctype, method, novalidate, target, submit 会导致表单提交混乱；

4、大小写，属性值

元素的标签和属性名必须小写, 属性值必须加双引号。

5、html模板

```
<!DOCTYPE HTML>
<html lang="zh-cmn-hans">
<head>
<meta charset="utf-8">
<meta name="renderer" content="webkit|ie-comp|ie-stand">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1,user-scalable=no" />
<title>title不可缺少，控制在25个字、50个字节以内。 “二级栏目 - 一级栏目 - 网站名称” 。</title>
<meta name="keywords" content="关键词,5个左右,单个8汉字以内">
<meta name="description" content="网站描述，字数尽量空制在80个汉字，160个字符以内！">
<link rel="Bookmark" href="/favicon.ico" >
<link rel="Shortcut Icon" href="/favicon.ico" />
<!--[if lt IE 9]>
<script>
      (function(){var e="abbr, article, aside, audio, canvas, datalist, details, dialog, eventsource, figure, f
ooter, header, hgroup, mark, menu, meter, nav, output, progress, section, time, video".split(', ');var i=e
.length;while(i--){document.createElement(e[i])}})()
   </script>
<![endif]-->
<link href="http://www.a.com/css/style.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="http://libs.baidu.com/jquery/1.8.2/jquery.js"></script>
</head>
<body>
<header>  <!--头部-->  </header>
<div>     <!--内容-->  </div>
<footer>  <!--头部-->  </footer>
<!--End of Footer-->
<script type="text/javascript" src="http://www.a.com/Lib/a.min.js"></script>

<!--下方是cnzz统计代码，请在自己项目中干掉-->
<script type="text/javascript">var cnzz_protocol= ...</script>
</body>
</html>
<!--@ChandlerVer5-->
```

CSS,SCSS开发规范

> 参考：
> http://alloyteam.github.io/CodeGuide/ (腾讯前端代码规范)
> http://www.waylau.com/css-code-guide/

1. 缩进

使用 soft tab（4个空格）。

```
.element {
    position: absolute;
    top: 10px;
    left: 10px;
    border-radius: 10px;
    width: 50px;
    height: 50px;
}
```

2. 分号

每个属性声明末尾都要加分号。

```
.element {
    width: 20px;
    height: 20px;
    background-color: red;
}
```

3. 空格

- 以下几种情况不需要空格：
  - 属性名后
  - 多个规则的分隔符`,`前
  - !important `!`后
  - 属性值中`(`后和`)`前
  - 行末不要有多余的空格
- 以下几种情况需要空格：
  - 属性值前
  - 选择器`>`, `+`, `~`前后
  - `{`前
  - !important `!`前
  - `@else`前后
  - 属性值中的`,`后
  - 注释`/*后和`*/`前

```
/* not good */
.element {
    color :red! important;
    background-color: rgba(0,0,0,.5);
}

/* good */
.element {
    color: red !important;
    background-color: rgba(0, 0, 0, .5);
}

/* not good */
.element ,
.dialog{
    ...
}

/* good */
.element,
.dialog {

}
```

```
}

/* not good */
.element>.dialog{
    ...
}

/* good */
.element > .dialog{
    ...
}

/* not good */
.element{
    ...
}

/* good */
.element {
    ...
}

/* not good */
@if{
    ...
}@else{
    ...
}

/* good */
@if {
    ...
} @else {
    ...
}
```

4. 空行

- 以下几种情况需要空行：
  - 文件最后保留一个空行
  - `}`后最好跟一个空行，包括scss中嵌套的规则
  - 属性之间需要适当的空行，具体见属性声明顺序

```
/* not good */
.element {
    ...
}
.dialog {
    color: red;
    &:after {
        ...
    }
}

/* good */
.element {
    ...
}

.dialog {
    color: red;

    &:after {
        ...
    }
}
```

5. 换行

- 以下几种情况不需要换行：
  - {`前
- 以下几种情况需要换行：
  - `{`后和`}`前
  - 每个属性独占一行
  - 多个规则的分隔符`,`后

```
/* not good */
.element
{color: red; background-color: black;}

/* good */
.element {
    color: red;
    background-color: black;
}

/* not good */
.element, .dialog {
    ...
}

/* good */
.element,
.dialog {
    ...
}
```

6. 注释

- 注释统一用 `/* */`（scss中也不要用 `//`），具体参照例子的写法；
- 缩进与下一行代码保持一致；
- 可位于一个代码行的末尾，与代码间隔一个空格。

```
/* Modal header */
.modal-header {
    ...
}

/*
 * Modal header
 */
.modal-header {
    ...
}

.modal-header {
    /* 50px */
    width: 50px;

    color: red; /* color red */
}
```

7. 引号

- 最外层统一使用双引号；

- url的内容要用引号；
- 属性选择器中的属性值需要引号。

```
.element:after {
    content: "";
    background-image: url("logo.png");
}

li[data-type="single"] {
    ...
}
```

8. 命名

- css 元素名称以及id和类名的命名也是区分大小写的（但是属性和值不区分）
- 类名使用小写字母，以中划线 `-` 分隔
- id采用驼峰式命名
- scss中的变量、函数、混合、placeholder采用驼峰式命名

```
/* class */
.element-content {
    ...
}

/* id */
#myDialog {
    ...
}

/* 变量 */
$colorBlack: #000;

/* 函数 */
@function pxToRem($px) {
    ...
}

/* 混合 */
@mixin centerBlock {
    ...
}

/* placeholder */
%myDialog {
    ...
}
```

9. 属性声明顺序

- 相关的属性声明按例子的顺序做分组处理，组之间需要有一个空行。

```
.declaration-order {
    display: block;
    float: right;

    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    z-index: 100;

    border: 1px solid #e5e5e5;
    border-radius: 3px;
    width: 100px;
    height: 100px;

    font: normal 13px "Helvetica Neue", sans-serif;
    line-height: 1.5;
    text-align: center;

    color: #333;
    background-color: #f5f5f5;

    opacity: 1;
}
// 下面是推荐的属性的顺序
[
    [
        "display",
        "visibility",
        "float",
        "clear",
        "overflow",
        "overflow-x",
        "overflow-y",
        "clip",
        "zoom"
    ],
    [
        "table-layout",
        "empty-cells",
        "caption-side",
        "border-spacing",
        "border-collapse",
        "list-style",
        "list-style-position",
        "list-style-type",
        "list-style-image"
    ],
    [
        "-webkit-box-orient",
        "-webkit-box-direction"
```

```
        "-webkit-box-direction",
        "-webkit-box-decoration-break",
        "-webkit-box-pack",
        "-webkit-box-align",
        "-webkit-box-flex"
    ],
    [
        "position",
        "top",
        "right",
        "bottom",
        "left",
        "z-index"
    ],
    [
        "margin",
        "margin-top",
        "margin-right",
        "margin-bottom",
        "margin-left",
        "-webkit-box-sizing",
        "-moz-box-sizing",
        "box-sizing",
        "border",
        "border-width",
        "border-style",
        "border-color",
        "border-top",
        "border-top-width",
        "border-top-style",
        "border-top-color",
        "border-right",
        "border-right-width",
        "border-right-style",
        "border-right-color",
        "border-bottom",
        "border-bottom-width",
        "border-bottom-style",
        "border-bottom-color",
        "border-left",
        "border-left-width",
        "border-left-style",
        "border-left-color",
        "-webkit-border-radius",
        "-moz-border-radius",
        "border-radius",
        "-webkit-border-top-left-radius",
        "-moz-border-radius-topleft",
        "border-top-left-radius",
        "-webkit-border-top-right-radius",
        "-moz-border-radius-topright",
        "border-top-right-radius",
        "-webkit-border-bottom-right-radius",
        "-moz-border-radius-bottomright",
        "border-bottom-right-radius",
```

```
        "-webkit-border-bottom-left-radius",
        "-moz-border-radius-bottomleft",
        "border-bottom-left-radius",
        "-webkit-border-image",
        "-moz-border-image",
        "-o-border-image",
        "border-image",
        "-webkit-border-image-source",
        "-moz-border-image-source",
        "-o-border-image-source",
        "border-image-source",
        "-webkit-border-image-slice",
        "-moz-border-image-slice",
        "-o-border-image-slice",
        "border-image-slice",
        "-webkit-border-image-width",
        "-moz-border-image-width",
        "-o-border-image-width",
        "border-image-width",
        "-webkit-border-image-outset",
        "-moz-border-image-outset",
        "-o-border-image-outset",
        "border-image-outset",
        "-webkit-border-image-repeat",
        "-moz-border-image-repeat",
        "-o-border-image-repeat",
        "border-image-repeat",
        "padding",
        "padding-top",
        "padding-right",
        "padding-bottom",
        "padding-left",
        "width",
        "min-width",
        "max-width",
        "height",
        "min-height",
        "max-height"
    ],
    [
        "font",
        "font-family",
        "font-size",
        "font-weight",
        "font-style",
        "font-variant",
        "font-size-adjust",
        "font-stretch",
        "font-effect",
        "font-emphasize",
        "font-emphasize-position",
        "font-emphasize-style",
        "font-smooth",
        "line-height",
        "text-align",
```

```
        "-webkit-text-align-last",
        "-moz-text-align-last",
        "-ms-text-align-last",
        "text-align-last",
        "vertical-align",
        "white-space",
        "text-decoration",
        "text-emphasis",
        "text-emphasis-color",
        "text-emphasis-style",
        "text-emphasis-position",
        "text-indent",
        "-ms-text-justify",
        "text-justify",
        "letter-spacing",
        "word-spacing",
        "-ms-writing-mode",
        "text-outline",
        "text-transform",
        "text-wrap",
        "-ms-text-overflow",
        "text-overflow",
        "text-overflow-ellipsis",
        "text-overflow-mode",
        "-ms-word-wrap",
        "word-wrap",
        "-ms-word-break",
        "word-break"
    ],
    [
        "color",
        "background",
        "filter:progid:DXImageTransform.Microsoft.AlphaImageLoader",
        "background-color",
        "background-image",
        "background-repeat",
        "background-attachment",
        "background-position",
        "-ms-background-position-x",
        "background-position-x",
        "-ms-background-position-y",
        "background-position-y",
        "-webkit-background-clip",
        "-moz-background-clip",
        "background-clip",
        "background-origin",
        "-webkit-background-size",
        "-moz-background-size",
        "-o-background-size",
        "background-size"
    ],
    [
        "outline",
        "outline-width",
```

```
        "outline-style",
        "outline-color",
        "outline-offset",
        "opacity",
        "filter:progid:DXImageTransform.Microsoft.Alpha(Opacity",
        "-ms-filter:\\'progid:DXImageTransform.Microsoft.Alpha",
        "-ms-interpolation-mode",
        "-webkit-box-shadow",
        "-moz-box-shadow",
        "box-shadow",
        "filter:progid:DXImageTransform.Microsoft.gradient",
        "-ms-filter:\\'progid:DXImageTransform.Microsoft.gradient",
        "text-shadow"
    ],
    [
        "-webkit-transition",
        "-moz-transition",
        "-ms-transition",
        "-o-transition",
        "transition",
        "-webkit-transition-delay",
        "-moz-transition-delay",
        "-ms-transition-delay",
        "-o-transition-delay",
        "transition-delay",
        "-webkit-transition-timing-function",
        "-moz-transition-timing-function",
        "-ms-transition-timing-function",
        "-o-transition-timing-function",
        "transition-timing-function",
        "-webkit-transition-duration",
        "-moz-transition-duration",
        "-ms-transition-duration",
        "-o-transition-duration",
        "transition-duration",
        "-webkit-transition-property",
        "-moz-transition-property",
        "-ms-transition-property",
        "-o-transition-property",
        "transition-property",
        "-webkit-transform",
        "-moz-transform",
        "-ms-transform",
        "-o-transform",
        "transform",
        "-webkit-transform-origin",
        "-moz-transform-origin",
        "-ms-transform-origin",
        "-o-transform-origin",
        "transform-origin",
        "-webkit-animation",
        "-moz-animation",
        "-ms-animation",
        "-o-animation",
        "animation",
```

```
        "-webkit-animation-name",
        "-moz-animation-name",
        "-ms-animation-name",
        "-o-animation-name",
        "animation-name",
        "-webkit-animation-duration",
        "-moz-animation-duration",
        "-ms-animation-duration",
        "-o-animation-duration",
        "animation-duration",
        "-webkit-animation-play-state",
        "-moz-animation-play-state",
        "-ms-animation-play-state",
        "-o-animation-play-state",
        "animation-play-state",
        "-webkit-animation-timing-function",
        "-moz-animation-timing-function",
        "-ms-animation-timing-function",
        "-o-animation-timing-function",
        "animation-timing-function",
        "-webkit-animation-delay",
        "-moz-animation-delay",
        "-ms-animation-delay",
        "-o-animation-delay",
        "animation-delay",
        "-webkit-animation-iteration-count",
        "-moz-animation-iteration-count",
        "-ms-animation-iteration-count",
        "-o-animation-iteration-count",
        "animation-iteration-count",
        "-webkit-animation-direction",
        "-moz-animation-direction",
        "-ms-animation-direction",
        "-o-animation-direction",
        "animation-direction"
    ],
    [
        "content",
        "quotes",
        "counter-reset",
        "counter-increment",
        "resize",
        "cursor",
        "-webkit-user-select",
        "-moz-user-select",
        "-ms-user-select",
        "user-select",
        "nav-index",
        "nav-up",
        "nav-right",
        "nav-down",
        "nav-left",
        "-moz-tab-size",
        "-o-tab-size",
```

```
        "tab-size",
        "-webkit-hyphens",
        "-moz-hyphens",
        "hyphens",
        "pointer-events"
    ]
]
```

10. 颜色

- 使颜色16进制用小写字母；
- 颜色16进制尽量用简写。

```
/* not good */
.element {
    color: #ABCDEF;
    background-color: #001122;
}

/* good */
.element {
    color: #abcdef;
    background-color: #012;
}
```

11. 属性简写

- 属性简写需要你非常清楚属性值的正确顺序，而且在大多数情况下并不需要设置属性简写中包含的所有值，所以建议尽量分开声明会更加清晰；
- margin 和 padding 相反，需要使用简写；
- 常见的属性简写包括：
  - font
  - background
  - transition
  - animation

```
/* not good */
.element {
    transition: opacity 1s linear 2s;
}

/* good */
.element {
    transition-delay: 2s;
    transition-timing-function: linear;
    transition-duration: 1s;
    transition-property: opacity;
}
```

12. 媒体查询

尽量将媒体查询的规则靠近与他们相关的规则，不要将他们一起放到一个独立的样式文件中，或者丢在文档的最底部，这样做只会让大家以后更容易忘记他们。

```
.element {
    ...
}

.element-avatar{
    ...
}

@media (min-width: 480px) {
    .element {
        ...
    }

    .element-avatar {
        ...
    }
}
```

13. 杂项

- 不允许有空的规则；
- 元素选择器用小写字母；
- 去掉小数点前面的0；
- 去掉数字中不必要的小数点和末尾的0；
- 属性值 `0` 后面不要加单位；
- 禁止使用 `expression` 表达式；
- 禁止滥用 `!important` ；
- 同个属性不同前缀的写法需要在垂直方向保持对齐，具体参照右边的写法；

- 无前缀的标准属性应该写在有前缀的属性后面；

```
-ms-transform:rotate(7deg);             //-ms代表ie内核识别码
-moz-transform:rotate(7deg);             //-moz代表火狐内核识别码
-webkit-transform:rotate(7deg);          //-webkit代表谷歌内核识别码
-o-transform:rotate(7deg);              //Opera15以前加 -o-，Opera15及以后加 -webkit-。
transform:rotate(7deg);                 //统一标识语句,符合w3c标准
```

- 不要在同个规则里出现重复的属性，如果重复的属性是连续的则没关系；
- 不要在一个文件里出现两个相同的规则；
- 用 `border: 0;` 代替 `border: none;` ；
- 在保证选择器准确的情况下尽量简化选择器 (尽量不超过4级，最多4级)，不写多余的选择器(如class/id前的标签选择器、li上一级的的ul/ol、dt/dd上一级的dl等，在保证准确的前提下能去掉的都去掉)（在 `scss` 中如果超过4层应该考虑用嵌套的方式来写）；
- 尽量少用 `*` 选择器。

```
/* not good */
.element {
}

/* not good */
LI {
    ...
}

/* good */
li {
    ...
}

/* not good */
.element {
    color: rgba(0, 0, 0, 0.5);
}

/* good */
.element {
    color: rgba(0, 0, 0, .5);
}

/* not good */
.element {
    width: 50.0px;
}

/* good */
.element {
    width: 50px;
}
```

```
/* not good */
.element {
    width: 0px;
}

/* good */
.element {
    width: 0;
}

/* not good */
.element {
    border-radius: 3px;
    -webkit-border-radius: 3px;
    -moz-border-radius: 3px;

    background: linear-gradient(to bottom, #fff 0, #eee 100%);
    background: -webkit-linear-gradient(top, #fff 0, #eee 100%);
    background: -moz-linear-gradient(top, #fff 0, #eee 100%);
}

/* good */
.element {
    -webkit-border-radius: 3px;
       -moz-border-radius: 3px;
            border-radius: 3px;

    background: -webkit-linear-gradient(top, #fff 0, #eee 100%);
    background:    -moz-linear-gradient(top, #fff 0, #eee 100%);
    background:         linear-gradient(to bottom, #fff 0, #eee 100%);
}

/* not good */
.element {
    color: rgb(0, 0, 0);
    width: 50px;
    color: rgba(0, 0, 0, .5);
}

/* good */
.element {
    color: rgb(0, 0, 0);
    color: rgba(0, 0, 0, .5);
}
```

- 避免使用 @import语句。如果使用了@import语句，一定要将它们写在样式表开头，保证它的优先级低于样式表中的其他规则；与 `<link>` 标签相比， `@import` 指令要慢很多，不光增加了额外的请求次数，还会导致不可预料的问题。替代办法有以下几种：
  - 使用多个 `<link>` 元素
  - 通过 Sass 或 Less 类似的 CSS 预处理器将多个 CSS 文件编译为一个文件
  - 通过 Rails、Jekyll 或其他系统中提供过 CSS 文件合并功能

- 请参考 http://www.stevesouders.com/blog/2009/04/09/dont-use-import/ 了解更多知识。

```
<!-- Use link elements -->
<link rel="stylesheet" href="core.css">

<!-- Avoid @imports -->
<style>
  @import url("more.css");
</style>
```

图片规范

图片规范

1. 所有页面元素类图片均放入img文件夹, 测试用图片放于img/demoimg文件夹;
2. 图片格式仅限于gif || png || jpg;
3. 命名全部用小写英文字母 || 数字 || _ 的组合，其中不得包含汉字 || 空格 || 特殊字符；尽量用易懂的词汇, 便于团队其他成员理解; 另, 命名分头尾两部分, 用下划线隔开, 比如ad_left01.gif || btn_submit.gif;
4. 在保证视觉效果的情况下选择最小的图片格式与图片质量, 以减少加载时间;
5. 尽量避免使用半透明的png图片(若使用, 请参考css规范相关说明);
6. 运用css sprite技术集中小的背景图或图标, 减小页面http请求, 但注意, 请务必在对应的sprite psd源图中划参考线, 并保存至img目录下.

HTML/CSS开发规范指南

> 参考：https://github.com/doyoe/html-css-guide

HTML/CSS开发规范指南

目录

规范概述

规范的制定是我们长期以来对工作的积累与沉淀的产物，帮助我们更快、更好、更高效的完成繁重、复杂、多样化的任务，我们制作规范的主要目的在于：

- 降低每个组员介入项目的门槛成本；
- 提高工作效率及协同开发的便捷性；
- 高度统一的代码风格；
- 提供一整套HTML、CSS解决方案，来帮助开发人员快速做出高质量的符合要求的页面。

基本信息

规范名称 | Cook

--------|------|

当前版本 | v1.0 beta

规范发起 | 杜瑶(@doyoe) 参与人群 | Qunar FED

最后更新 | 2015.03.20

通用约定

1.文档目录结构

```
|-- 项目名
    |-- src 开发环境
        |-- html    静态页面模板目录
        |-- bgimg   背景图目录（假设有的话）
        |-- image   前景图目录（假设有的话）
        |-- font    字体目录（假设有的话）
        |-- scripts     脚本目录
        |-- styles（Yo） 样式目录
            |-- lib   基础库
                |-- core     核心代码：reset
                |-- element   元素
                |-- fragment  公用碎片
                |-- layout    布局
                |-- widget    组件
            |-- usage  项目具体实现
                |-- project 某个子项目
                    |-- core     核心代码：桥接lib中的core，可以进行项目级扩展
                    |-- fragment  项目公用碎片
                    |-- module    模块
                    |-- page      page桥接文件目录：src-list
                    |-- export    page pack之后的文件目录
    |-- prd 生产环境
        |-- bgimg   背景图目录（假设有的话）
        |-- image   前景图目录（假设有的话）
        |-- font    字体目录（假设有的话）
        |-- scripts     脚本目录
        |-- styles（Yo） 样式目录
            |-- project1 子项目
                |-- index.css
                |-- login.css
                |-- and etc...
            |-- project2 子项目
                |-- index.css
                |-- login.css
                |-- and etc...
            |-- and etc...
```

> `src`，`scripts`，`styles` 三个目录是为了和现有项目保持一致，避免修改过大，所以保持不变。
>
> `html` 目录，用于存放前端开发做的静态页面，以备查阅、备份、review或给后端套页面。

> bgimg，image，font 三个目录在Qunar一般不会直接存在，因为我们有source服务器，这些资源都会在那上面管理；不过特殊情况也会有，比如一些独立的项目，没有使用source的，那么就需要遵循这样的目录划分。
>
> 至于 html，bgimg，image，font 这几个目录为什么没有加 s，主要是因为不希望大家去想某个目录是否为复数，简单点就好。
>
> prd 为生产环境目录，以 xxx 项目中的一个子项目 mobile 为例，其生产环境中的某个CSS外链大致如下：//sitename.com/prd/styles/mobile/index.css

2.分离

结构（HTML）、表现（CSS）、行为分离（JavaScript）

> 将结构与表现、行为分离，保证它们之间的最小耦合，这对前期开发和后期维护都至关重要。

3.文件命名

- CSS模块文件，其文件名必须与模块名一致；

假定有这样一个模块：

```
.m-detail { sRules; }
.m-detail-hd { sRules; }
.m-detail-bd { sRules; }
.m-detail-ft { sRules; }
```

> 那么该模块的文件名应该为：m-detail.css

- CSS页面文件，其文件名必须与HTML文件名一致；

> 假定有一个HTML页面叫 product.html，那么其相对应的CSS页面文件名应该为：product.css
>
> 假定现在有一个 product.html，里面有2个模块：

```
+<section class="m-list">
+<section class="m-info">
```

那么开发人员能快速找到与该页面相关的3个直接CSS文件，包括：product.css，m-list.css，m-info.css

4.缩进

使用tab（4个空格宽度）来进行缩进，可以在IDE里进行设置

5.编码

- 以 UTF-8 无 BOM 编码作为文件格式；
- 在HTML中文档中用 `<meta charset="utf-8" />` 来指定编码；
- 为每个CSS文档显示的定义编码，在文档首行定义 `@charset "utf-8";`

> 在 Sass 中，如果文档中出现中文，却未显示定义编码，将会编译出错，为了统一各种写法，且提前规避错误几率，统一要求每个CSS文档都需要定义编码

6.小写

- 所有的HTML标签必须小写；
- 所有的HTML属性必须小写；
- 所有的样式名及规则必须小写。

7.注释

尽可能的为你的代码写上注释。解释为什么要这样写，它是新鲜的方案还是解决了什么问题。

8.待办事项

用 TODO 标示待办事项和正在开发的条目

```
<!-- TODO: 图文混排 -->
<div class="g-imgtext">
<img src="1.png" alt="" />
...

/* TODO: 图文混排 comm: g-imgtext */
.g-imgtext { sRules; }
```

9.行尾空格

删除行尾空格，这些空格没有必要存在

10.省略嵌入式资源协议头

省略图像、媒体文件、样式表和脚本等URL协议头部声明（http:，https:）。如果不是这两个声明的URL则不省略。

省略协议声明，使URL成相对地址，防止内容混淆问题和导致小文件重复下载（这个主要是指http和https交杂的场景中）。

不推荐：

```
<script src="http://www.google.com/js/gweb/analytics/autotrack.js"></script>
```

推荐：

```
<script src="//www.google.com/js/gweb/analytics/autotrack.js"></script>
```

不推荐：

```
.example {
  background: url(http://www.google.com/images/example);
}
```

推荐：

```
.example {
  background: url(//www.google.com/images/example);
}
```

> 注：省略协议头在IE7-8下会有一点小问题，外部CSS文件（link和@import）会被下载两遍，所以该条目的约定看具体项目。

11.代码有效性

- 使用 W3C HTML Validator 来验证你的HTML代码有效性；
- 使用 W3C CSS Validator 来验证你的CSS代码有效性。

> 代码验证不是最终目的，真的目的在于让开发者在经过多次的这种验证过程后，能够深刻理解到怎样的语法或写法是非标准和不推荐的，即使在某些场景下被迫要使用非标准写法，也可以做到心中有数。

HTML约定

1.文档类型

- 统一使用HTML5的标准文档类型：`<!DOCTYPE html>`；

> HTML5文档类型具备前后兼容的特质，并且易记易书写

- 在文档doctype申明之前，不允许加上任何非空字符；

> 任何出现在doctype申明之前的字符都将使得你的HTML文档进入非标准模式

- 不允许添加 `<meta>` 标签强制改变文档模式。

> 避免出现不可控的问题

2.省略type属性

在调用CSS和JavaScript时，可以将type属性省略不写

不允许：

```
<link type="text/css" rel="stylesheet" href="base.css" />
<script type="text/javascript" src="base.js"></script>
```

应该：

```
<link rel="stylesheet" href="base.css" />
<script src="base.js"></script>
```

> 因为HTML5在引入CSS时，默认type值为text/css；在引入JavaScript时，默认type值为text/javascript

3.省略属性值

非必须属性值可以省略

不允许：

```
<input type="text" readonly="readonly" />
<input type="text" disabled="disabled" />
```

应该：

```
<input type="text" readonly />
<input type="text" disabled />
```

> 这里的 readonly 和 disabled 属性的值是非必须的，可以省略不写，我们知道HTML5表单元素新增了很多类似的属性，如: required

4.用双引号包裹属性值

所有的标签属性值必须要用双引号包裹，同时也不允许有的用双引号，有的用单引号的情况

不允许：

```
<a href=http://www.qunar.com class=home>去哪儿网</a>
```

应该：

```
<a href="http://www.qunar.com" class="home">去哪儿网</a>
```

5.嵌套

所有元素必须正确嵌套

- 不允许交叉；

不允许：

```
<span><dfn>交叉嵌套</span></dfn>
```

应该：

```
<span><dfn>交叉嵌套</dfn></span>
```

- 不允许非法的子元素嵌套。

不允许：

```
<ul>
   <h3>xx列表</h3>
   <li>asdasdsdasd</li>
   <li>asdasdsdasd</li>
</ul>
```

应该：

```
<div>
   <h3>xx列表</h3>
   <ul>
      <li>asdasdsdasd</li>
      <li>asdasdsdasd</li>
   </ul>
</div>
```

- 不推荐inline元素包含block元素；

不推荐：

```
<span>
  <h1>这是一个块级h1元素</h1>
  <p>这是一个块级p元素</p>
</span>
```

推荐：

```
<div>
  <h1>这是一个块级h1元素</h1>
  <p>这是一个块级p元素</p>
</div>
```

> 规则可参考：
>
> HTML4/XHTML1.0 Strict: 嵌套规则。
>
> HTML5: 嵌套规则
>
> 举个例子，在HTML5中，a元素同时属于 Flow content, Phrasing content, Interactive content, Palpable content 4个分类，那些子元素是 phrasing 元素的元素可以是 a 的父元素，a 允许的子元素是以它的父元素允许的子元素为准，但不能包含 interactive 元素。

6.标签闭合

所有标签必须闭合

不允许：

```
<div>balabala...
<link rel="stylesheet" href="*.css">
```

应该：

```
<div>balabala...</div>
<link rel="stylesheet" href="*.css" />
```

> 虽然有些标记没有要求必须关闭，但是为了避免出错的几率，要求必须全部关闭，省去判断某标记是否需要关闭的时间

7.多媒体替代方案

- 为img元素加上alt属性；
- 为视频内容提供音轨替代；
- 为音频内容提供字母替代等等。

不推荐：

```
<img src="banner.jpg" />
```

推荐：

```
<img src="banner.jpg" alt="520即将到来，爱就大声说出来" />
```

> alt属性的内容为对该图片的简要描述，这对于盲人用户和图像损毁都非常有意义，即无障碍。对于纯粹的装饰性图片，alt属性值可以留空，如 alt=""

8.有效操作

为表单元素label加上for属性

不允许：

```
<input type="radio" name="color" value="0" /><label>蓝色</label>
<input type="radio" name="color" value="1" /><label>粉色</label>
```

应该：

```
<input type="radio" id="blue" name="color" value="0" /><label for="blue">蓝色</label>
<input type="radio" id="pink" name="color" value="1" /><label for="pink">粉色</label>
```

> for属性能让点击label标签的时候，同时focus到对应的 input 和 textarea上，增加响应区域

9.按模块添加注释

在每个模块开始和结束的地方添加注释

```
<!-- 新闻列表模块 -->
<div class="m-news g-mod"
...
<!-- /新闻列表模块 -->

<!-- 排行榜模块 -->
<div class="m-topic g-mod"
...
<!-- /排行榜模块 -->
```

> 注释内容左右两边保留和注释符号有1个空格位，在注释内容内不允许再出现中划线“-”，某些浏览器会报错。

> 注释风格保持与原生HTML的语法相似：成对出现 `<!-- comment --><!-- /comment -->`

10.格式

- 将每个块元素、列表元素或表格元素都放在新行；
- inline元素视情况换行，以长度不超过编辑器一屏为宜；
- 每个子元素都需要相对其父级缩进（参见缩进约定）。

不推荐：

```
<div><h1>asdas</h1><p>dff<em>asd</em>asda<span>sds</span>dasdasd</p></div>
```

推荐：

```
<div>
  <h1>asdas</h1>
  <p>dff<em>asd</em>asda<span>sds</span>dasdasd</p>
</div>
```

11.语义化标签

- 根据HTML元素的本身用途去使用它们；
- 禁止使用被废弃的用于表现的标签，如 center, font 等；
- 部分在XHTML1中被废弃的标签，在HTML5中被重新赋予了新的语义，在选用时请先弄清其语义，如:b, i, u等。

不允许：

```
<p>标题</p>
```

应该：

```
<h1>标题</h1>
```

> 虽然使用p标签，也可以通过CSS去定义它的外观和标题相同，但p标签本身的并不是表示标题，而是表示文本段落
>
> 参阅：HTML5 Elements

12.模块化

- 每个模块必须有一个模块名；
- 每个模块的基本组成部分应该一致；

- 模块的子节点类名需带上模块名（防止模块间嵌套时产生不必要的覆盖）；
- 孙辈节点无需再带模块名。

代码如：

```
<section class="m-detail">
    <header class="m-detail-hd">
        <h1 class="title">模块标题</h1>
    </header>
    <div class="m-detail-bd">
        <p class="info">一些实际内容</p>
    </div>
    <footer class="m-detail-ft">
        <a href="#" class="more">更多</a>
    </footer>
</section>
```

> 其中 `.m-detail-hd`，`.m-detail-bd`，`.m-detail-ft` 为可选，视具体模块情况决定是否需要抽象为这种头，中，尾 的结构

CSS约定

1.文件引用

- 一律使用link的方式调用外部样式
- 不允许在页面中使用 `<style>` 块；
- 不允许在 `<style>` 块中使用 `@import`；
- 不允许使用 `style` 属性写行内样式。

> 一般情况下，在页面中只允许使用 `<link />` 标签来引用CSS文件，

2.命名-组成元素

- 命名必须由单词、中划线①或数字组成；
- 不允许使用拼音（约定俗成的除外，如：youku, baidu），尤其是缩写的拼音、拼音与英文的混合。

不推荐：

```
.xiangqing { sRules; }
.news_list { sRules; }
.zhuti { sRules; }
```

推荐：

```
.detail { sRules; }
.news-list { sRules; }
.topic { sRules; }
```

> ①我们使用中划线 “-” 作为连接字符，而不是下划线 "_"。
>
> 我们知道2种方式都有不少支持者，但 "-" 能让你少按一次shift键，并且更符合CSS原生语法，所以我们只选一种目前业内普遍使用的方式

3.命名-词汇规范

- 不依据表现形式来命名；
- 可根据内容来命名；
- 可根据功能来命名。

不推荐：

```
left, right, center, red, black
```

推荐：

```
nav, aside, news, type, search
```

4.命名-缩写规范

- 保证缩写后还能较为清晰保持原单词所能表述的意思；
- 使用业界熟知的或者约定俗成的。

不推荐：

```
navigation   =>  navi
header       =>  head
description  =>  des
```

推荐：

```
navigation   =>  nav
header       =>  hd
description  =>  desc
```

5.命名-前缀规范

前缀|说明|示例

---|---|---|
g-|全局通用样式命名，前缀g全称为global，一旦修改将影响全站样式|g-mod
m-|模块命名方式|m-detail
ui-|组件命名方式|ui-selector
js-|所有用于纯交互的命名，不涉及任何样式规则。JSer拥有全部定义权限|js-switch

- 选择器必须是以某个前缀开头

不推荐：

```
.info { sRules; }
.current { sRules; }
.news { sRules; }
```

> 因为这样将给我们带来不可预知的管理麻烦以及沉重的历史包袱。你永远也不会知道哪些样式名已经被用掉了，如果你是一个新人，你可能会遭遇，你每定义个样式名，都有同名的样式已存在，然后你只能是换样式名或者覆盖规则。

推荐：

```
.m-detail .info { sRules; }
.m-detail .current { sRules; }
.m-detail .news { sRules; }
```

> 所有的选择器必须是以 g-, m-, ui- 等有前缀的选择符开头的，意思就是说所有的规则都必须在某个相对的作用域下才生效，尽可能减少全局污染。

js- 这种级别的className完全交由JSer自定义，但是命名的规则也可以保持跟重构一致，比如说不能使用拼音之类的

6.id与class

重构工程师只允许使用class（因历史原因及大家的习惯做出妥协）。

7.书写格式

- 选择器与大括号之间保留一个空格；
- 分号之后保留一个空格；
- 逗号之后保留一个空格；
- 所有规则需换行；
- 多组选择器之间需换行。

不推荐：

```
main{
    display:inline-block;
}
h1,h2,h3{
    margin:0;
    background-color:rgba(0,0,0,.5);
}
```

推荐：

```
main {
    display: inline-block;
}
h1,
h2,
h3 {
    margin: 0;
    background-color: rgba(0, 0, 0, .5);
}
```

8.规则与分号

每条规则结束后都必须加上分号

不推荐：

```
body {
    margin: 0;
    padding: 0;
    font-size: 14px
}
```

推荐：

```
body {
    margin: 0;
    padding: 0;
    font-size: 14px;
}
```

9.0与单位

如果属性值为0，则不需要为0加单位

不推荐：

```
body {
    margin: 0px;
    padding: 0px;
}
```

推荐：

```
body {
    margin: 0;
    padding: 0;
}
```

10.0与小数

如果是0开始的小数，前面的0可以省略不写

不推荐：

```
body {
    opacity: 0.6;
    text-shadow: 1px 1px 5px rgba(0, 0, 0, 0.5);
}
```

推荐：

```
body {
    opacity: .6;
    text-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
}
```

11.去掉uri中引用资源的引号

不要在url()里对引用资源加引号

不推荐：

```
body {
    background-image: url("sprites.png");
}
@import url("global.css");
```

推荐：

```
body {
    background-image: url(sprites.png);
}
@import url(global.css);
```

12.HEX颜色值写法

- 将所有的颜色值小写；
- 可以缩写的缩写至3位。

不推荐：

```
body {
    background-color: #FF0000;
}
```

推荐：

```
body {
    background-color: #f00;
}
```

13.属性书写顺序

- 遵循先布局后内容的顺序。

```
.g-box {
        display: block;
        float: left;
        width: 500px;
        height: 200px;
        margin: 10px;
        padding: 10px;
        border: 10px solid;
        background: #aaa;
        color: #000;
        font: 14px/1.5 sans-serif;
}
```

> 这个应该好理解，比如优先布局，我们知道布局属性有 display, float, overflow 等等；内容次之，比如 color, font, text-align 之类。

- 组概念。

拿上例的代码来说，如果我们还需要进行定位及堆叠，规则我们可以改成如下：

```
.g-box {
        display: block;
        position: relative;
        z-index: 2;
        top: 10px;
        left: 100px;
        float: left;
        width: 500px;
        height: 200px;
        margin: 10px;
        padding: 10px;
        border: 10px solid;
        background: #aaa;
        color: #000;
        font: 14px/1.5 sans-serif;
}
```

> 从代码中可以看到，我们直接将z-index, top, left 紧跟在 position 之后，因为这几个属性其实是一组的，如果去掉position，则后3条属性规则都将失效。

- 私有属性在前标准属性在后

```
.g-box {
        -webkit-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
        -moz-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
        -o-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
        box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
}
```

> 当有一天你的浏览器升级后，可能不再支持私有写法，那么这时写在后面的标准写法将生效，避免无法向后兼容的情况发生。

14.注释规范

保持注释内容与星号之间有一个空格的距离

普通注释（单行）

```
/* 普通注释 */
```

区块注释

```
/**
 * 模块: m-detail
 * 描述：酒店详情模块
 * 应用：page detail, info and etc...etc
 */
```

> 有特殊作用的规则一定要有注释说明
> 应用了高级技巧的地方一定要注释说明

15.hack规范

- 尽可能的减少对Hack的使用和依赖，如果在项目中对Hack的使用太多太复杂，对项目的维护将是一个巨大的挑战；
- 使用其它的解决方案代替Hack思路；
- 如果非Hack不可，选择稳定且常用并易于理解的。

```
.test {
        color: #000;       /* For all */
        color: #111\9;     /* For all IE */
        color: #222\0;     /* For IE8 and later, Opera without Webkit */
        color: #333\9\0;   /* For IE8 and later */
        color: #444\0/;    /* For IE8 and later */
        *color: #666;      /* For IE7 and earlier */
        _color: #777;      /* For IE6 and earlier */
}
```

- 严谨且长期的项目，针对IE可以使用条件注释作为预留Hack或者在当前使用

IE条件注释语法：

```
<!--[if <keywords>? IE <version>?]>
<link rel="stylesheet" href="*.css" />
<![endif]-->
```

语法说明：

```
<keywords>
if条件共包含6种选择方式：是否、大于、大于或等于、小于、小于或等于、非指定版本
是否：指定是否IE或IE某个版本。关键字：空
大于：选择大于指定版本的IE版本。关键字：gt（greater than）
大于或等于：选择大于或等于指定版本的IE版本。关键字：gte（greater than or equal）
小于：选择小于指定版本的IE版本。关键字：lt（less than）
小于或等于：选择小于或等于指定版本的IE版本。关键字：lte（less than or equal）
非指定版本：选择除指定版本外的所有IE版本。关键字：!
```

<version>
目前的常用IE版本为6.0及以上，推荐酌情忽略低版本，把精力花在为使用高级浏览器的用户提供更好的体验上，另从IE10开始已无此特性

16.避免低效率选择器

- 避免类型选择器

不允许：

```
div#doc { sRules; }
li.first { sRules; }
```

应该：

```
#doc { sRules; }
.first { sRules; }
```

> CSS选择器是由右到左进行解析的，所以 div#doc 本身并不会比 #doc 更快

- 避免多id选择器

不允许：

```
#xxx #yyy { sRules; }
```

应该：

```
#yyy { sRules; }
```

17.属性缩写与分拆

- 无继承关系时，使用缩写

不推荐：

```
body {
        margin-top: 10px;
        margin-right: 10px;
        margin-bottom: 10px;
        margin-left: 10px;
}
```

推荐：

```
body {
        margin: 10px;
}
```

- 存在继承关系时，使用分拆方式

不推荐：

```
.m-detail {
        font: bold 12px/1.5 arial, sans-serif;
}
.m-detail .info {
        font: normal 14px/1.5 arial, sans-serif;
}
```

要避免错误的覆盖：

```
.m-detail .info {
        font: 14px sans;
}
```

> 如果你只是想改字号和字体，然后写成了上面这样，这是错误的写法，因为 `font` 复合属性里的其他属性将会被重置为 user agent 的默认值，比如 `font-weight` 就会被重置为 `normal`。

推荐：

```
.m-detail {
        font: bold 12px/1.5 arial, sans-serif;
}
.m-detail .info {
        font-weight: normal;
        font-size: 14px;
}
```

> 在存在继承关系的情况下，只将需要变更的属性重定义，不进行缩写，避免不需要的重写的属性被覆盖定义

- 根据规则条数选择缩写和拆分

不推荐：

```
.m-detail {
        border-width: 1px;
        border-style: solid;
        border-color: #000 #000 #f00;
}
```

推荐：

```
.m-detail {
        border: 1px solid #000;
        border-bottom-color: #f00;
}
```

18.模块化

- 每个模块必须是一个独立的样式文件，文件名与模块名一致；
- 模块样式的选择器必须以模块名开头以作范围约定；

假定有一个模块如前文 HTML模块化，那么 `m-detail.scss` 的写法大致如下：

```
.m-detail {
    background: #fff;
    color: #333;
    &-hd {
        padding: 5px 10px;
        background: #eee;
        .title {
            background: #eee;
        }
    }
    &-bd {
        padding: 10px;
        .info {
            font-size: 14px;
            text-indent: 2em;
        }
    }
    &-ft {
        text-align: center;
        .more {
            color: blue;
        }
    }
}
```

编译之后代码如下：

```
.m-detail {
    background: #fff;
    color: #333;
}
.m-detail-hd {
    padding: 5px 10px;
    background: #eee;
}
.m-detail-hd .title {
    background: #eee;
}
.m-detail-bd {
    padding: 10px;
}
.m-detail-bd .info {
    font-size: 14px;
    text-indent: 2em;
}
.m-detail-ft {
    text-align: center;
}
.m-detail-ft .more {
    color: blue;
}
```

> 任何超过3级的选择器，需要思考是否必要，是否有无歧义的，能唯一命中的更简短的写法

图像约定

1.图像压缩

所有图片必须经过一定的压缩和优化才能发布

2.背景图

- 使用PNG格式而不是GIF格式，因为PNG格式色彩更丰富，还能提供更好的压缩比；
- 在需要兼容IE6的项目中，尽可能选择PNG8，而不是使用PNG24+滤镜。

3.前景图

- 内容图片建议使用JPG，可以拥有更好地显示效果；
- 装饰性图片使用PNG。

4.Sprite

- CSS Sprite是一种将数个图片合成为一张大图的技术（既可以是背景图也可以是前景图），然后通过偏移来进行图像位置选取；

- CSS Sprite可以减少http请求。

结语

坚持一致性的原则。

一个团队的代码风格如果统一了，首先可以培养良好的协同和编码习惯，其次可以减少无谓的思考，再次可以提升代码质量和可维护性。

统一的代码风格，团队内部阅读或编辑代码，将会变得非常轻松，因为所有组员都处在一致思维环境中。

JavaScript

1、变量命名只允许有字母、数字、 $ 和 _ 下划线。

2。javascript 的变量和其他语法元素名都是区分大小写。

3。使用jQuery对象时候，使用$var的形式，让人很明白这是jQuery的对象（虽然有些库也会使用$），*还有W3school也不推荐*【变量也能以 $ 和 _ 符号开头（不过我们不推荐这么做）】

4、Google JavaScript代码风格指南 5、浅谈 JavaScript 编程语言的编码规范

遵行惯用法:

1. 注释符号 // 后应该空一格；
2. 防止变量提升，应先声明后使用（JSHint 会提醒出 _height 存在变量提升以及定义后未使用的错误）；
3. 不应该使用硬编码，并且重复几次（ ID 后缀名可以定义到常量里，用大写字母）；
4. 不应该有两个配置属性，含义不明（this.opts 和 this._options）；
5. 若两次以上引用同一对象的属性，应该定义到局部变量再引用（var options = this._options）；
6. 不应该同时使用两种属性命名风格（colModel 和 table_body）；
7. 局部变量名应该尽可能短，而方法名应该尽可能完整（不应该同时即有 fromatTpl 又有 parseTemplate）；
8. 局部变量名不需要用下划线开头，仅对象私有属性和私有方法有此必要；变量名不需要带类型属性（_thdoms 叫 ths 就好）；
9. 使用 JavaScript 时，for 循环基本可以避免（比如 jQuery 有 $.each, $.map，$.filter, $.grep 等等高阶函数可用）；
10. jQuery 对象名习惯以 $ 开头，以便区分 DOM 对象；jQuery 查询应尽量使用 context （如 this.$table = $('table', this.$element) ）；
11. jQuery DOM 操作和原生 DOM 操作不应该混用（已经使用 jQuery 的情况，就应该坚持使用 jQuery 来操作 DOM，避免丑陋的原生操作）；
12. DOM 元素构造出来，也不应该再到文档中查询一遍了（图上的构造太复杂，一眼真看不懂）；

Code Review把程序写正确还只是跨出了第一步。把代码交给你的同事和朋友 review，这是学习经验、共同提高 最快的办法。

注释规范

1、注释规范

目前脚本、样式的注释格式都有一个已经成文的约定规范，最初是YUI Compressor制定。

```
/**
 * 这里的注释内容【会】被压缩工具压缩
 */

/*!
 * 这里的注释内容【不会】被压缩工具压缩
 * 与上面一个注释块不同的是，第2个*换成了!
 */
```

其中说到这里说到的压缩工具有YUI Compressor 、Google Closure Compiler、gulp-uglify、grunt-contrib-uglify等，这些压缩工具都支持以上的压缩约定。常常把文件的关键信息放在第2种注释内容里，如文件名称、版本号、作者等。

关于这些关键信息，都由一些关键词和一定的格式来书写。关键词书写格式为：
使用 @key desc 格式来书写，常用的关键词有：

关键词	描述
@auhor	作者
@param	参数
@example	示例
@link	链接
@namespace	命名空间
@requires	依赖模块
@return	返回值
@version	版本号

其中，param关键词的格式为：

```
/**
 * @param {String} 参数描述
 */
```

> 参考资料
> YUI Compressor注释规范：http://yui.github.io/yuidoc/syntax/ JSDOC：http://usejsdoc.org/
> FED文章：http://frontenddev.org/article/sublime-does-text-3-plug-in-docblockr-with-javascript-comments-specification.html

编程注意

1、分号
分号表示语句的结束。大多数情况下，如果你省略了句尾的分号，Javascript会自动添加。

> 不要省略句末的分号。

2、with语句
with可以减少代码的书写，但是会造成混淆。

> 不要使用with语句。

3、相等和严格相等
Javascript有两个表示"相等"的运算符："相等"（==）和"严格相等"（===）。

> 不要使用"相等"（==）运算符，只使用"严格相等"（===）运算符。

4、全局变量
Javascript最大的语法缺点，可能就是全局变量对于任何一个代码块，都是可读可写。这对代码的模块化和重复使用，非常不利。

> 避免使用全局变量；如果不得不使用，用大写字母表示变量名，比如UPPER_CASE。

5、Javascript会自动将变量声明"提升"（hoist）到代码块（block）的头部。

> 所有变量声明都放在函数的头部。
> 规则10：所有函数都在使用之前定义。

6、自增和自减运算符
自增（++）和自减（--）运算符，放在变量的前面或后面，返回的值不一样，很容易发生错误。
事实上，所有的++运算符都可以用"+= 1"代替。

> 不要使用自增（++）和自减（--）运算符，用+=和-=代替。

7、区块
如果循环和判断的代码体只有一行，Javascript允许该区块（block）省略大括号。

> 总是使用大括号表示区块。

8、new命令
Javascript使用new命令，从建构函数生成一个新对象。

`var o = new myObject();` 这种做法的问题是，一旦你忘了加上new，myObject()内部的this关键字就会指向全局对象，导致所有绑定在this上面的变量，都变成全部变量。

如果不得不使用new，为了防止出错，最好在视觉上把建构函数与其他函数区分开来。

> 不要使用new命令，改用Object.create()命令。
> 建构函数的函数名，采用首字母大写（InitialCap）；其他函数名，一律首字母小写。

最流行的JavaScript代码规范

什么是最佳的JavaScript代码编程规范？这可能是一个众口难调的问题。那么，不妨换个问题，什么代码规范最流行？
sideeffect.kr通过分析GitHub上托管的开源代码，得出了一些有趣的结果。一起来看看吧。

行末逗号：

```
var foo = 1,
    bar = 2,
    baz = 3;

var obj = {
    foo: 1,
    bar: 2,
    baz: 3
};
```

空格缩进：

> 这年头大家都爱用空格了。使用空格缩进可以保证不同的开发者、不同的编辑器设置下看到的结果是一样的。
>
> 空格，81.1 %；Tab，18.9 %。（基于2,019,550次提交统计。）

函数名称后无空格：

```
function foo() {
  return "bar";
}
```

函数参数与括号间无空格：

```
function fn(arg1, arg2) {
//or
if (true) {
```

对象字面量的冒号后加空格，冒号前不加：

```
{
  foo: 1,
  bar: 2,
  baz: 3
}
```

条件语句关键字后加空格：

```
if (true) {
  //...
}

while (true) {
  //...
}

switch (v) {
  //...
}
```

单引号、双引号：

单引号，56.791 %；双引号，43.209 %。（基于1,705,910次提交。）

JavaScript 编程风格

PHP

> 参考：PHP 开发规范

技术部php开发规范将参照PEAR的规范，基本采用PEAR指定的规范，在其基础上增加、修改或删除部分适合具体开发环境的规范。

本规范包含了PHP开发时程序编码中命名规范、代码缩进规则、控制结构、函数调用、函数定义、注释、包含代码、PHP标记、文件头的注释块、URL样例、常量命名等方面的规则。

在开头就提到编码规范的目的是提高工作效率，保证开发的有效性和合理性，并最大程度提高程序代码的可读性和可重复利用性，减少返工，提高沟通效率。好的代码风格，见代码如见人，读代码时就好像和作者本人在亲切交谈。

PHP是一门脚本语言，用于网站开发，和JavaScript类似，本身就是一门不严谨的语言，有人评价PHP是“dirty and fast language”（脏而快的开发语言），所以规范显得更重要。

最后要说明的是，本规范不是强制，也不是标准。“约定大于规范”，如果有的规范太死板，不适应你所在团队，你可以不采用，而是按照自己的规范进行。

强烈建议使用大型的IDE进行代码开发和管理，并使用合适的版本控制软件。

书写规则

1、缩进

使用4个空格作为缩进，而不使用tab缩进。

4个空格常被作为缩进排版的一个单位。缩进的确切解释并未详细指定(空格 vs. 制表符)。

一个制表符等于8个空格(而非4个)，所以在某些编辑器中，需要特别指定一下制表符的长度为4（UltraEdit），而在某些编辑器中，会将制表符转换为空格。

2、大括号{}书写规则

{ 直接跟在控制语句之后，不换行，如：

```
for ( $i=0;$i<$count;$i++ ) {
    echo 'test';
}
```

3、小括号()和函数、关键词等

小括号、关键词和函数遵循以下规则：

a．不要把小括号和关键词紧贴在一起，要用一个空格间隔；如 if ($a<$b) ；

b．小括号和函数名间没有空格；如 $test = date("ymdhis") ；

c．除非必要，不要在 return 返回语句中使用小括号。 如 return $a ； 。

4、=符号书写

在程序中=符号的书写遵循以下规则：

a．在=符号的两侧，均需留出一个空格；如 $a = $b 、 $a = 'test' 等；

b．在=符号与!、=、<、>等符号相邻时，不需留一个空格；如 if ($a == $b) 、 if ($a != $b) 等；

c．在一个申明块，或者实现同样功能的一个块中，要求=号尽量上下对其，左边可以为了保持对齐使用多个空格，而右边要求空一个空格；如下例：

```
$testa   = $aaa;
$testaa  = $bbb;
$testaaa = $ccc;
```

5、if else swith for while等书写

对于控制结构的书写遵循以下规则：

a．在if条件判断中，如果用到常量判断条件，将常量放在等号或不等号的左边，例如： if (6 == $errorNum) ,因为如果你在等式中漏了一个等号，语法检查器会为你报错，可以很快找到错误位置，这样的写法要注意；

b． switch 结构中必须要有 default 块；

c．在 for 和 wiile 的循环使用中，要警惕 continue 、 break 的使用，避免产生类似 goto 的问题。

6、语句断行

在代码书写中，遵循以下原则：

a．尽量保证程序语句一行就是一句，而不要让一行语句太长产生折行；

b．尽量不要使一行的代码太长，一般控制在120个字符以内；

c．如果一行代码太长，请使用类似 .= 的方式断行书写；

d．对于执行数据库的sql语句操作，尽量不要在函数内写sql语句，而先用变量定义sql语句，然后在执行操作的函数中调用定义的变量。

例子：

```
$sql = 'SELECT username,password,address,age,postcode FROM test_t ';
$sql .= ' WHERE username=\'aaa\'';
$res = mysql_query($sql);
```

7、数字

一个在源代码中使用了的赤裸裸的数字是不可思议的数字，因为包括作者，在三个月内，没人知道它的含义。例如：

```
if ( 22 == $foo ) {
    start_thermo_nuclear_war();
} elseif ( 19 == $foo){
    refund_lotso_money();
} else {
    cry_cause_in_lost();
}
```

你应该用 define() 来给你想表示某样东西的数值一个真正的名字，而不是采用赤裸裸的数字，例如：

```
define('PRESIDENT_WENT_CRAZY', '22');
define('WE_GOOFED', '19');
define('THEY_DIDNT_PAY', '16');
if ( PRESIDENT_WENT_CRAZY == $foo ) {
    start_thermo_nuclear_war();
} elseif ( WE_GOOFED == $foo){
    refund_lotso_money();
} elseif ( THEY_DIDNT_PAY == $foo ){
    infinite_loop();
} else {
    cry_cause_in_lost();
}
```

8、判断

遵循以下规则：

a．不能使用 1/0 代替 true/false ，在 PHP 中，这是不相等的；

b．不要使用非零的表达式、变量或者方法直接进行 true/false 判断，而必须使用严格的完整 true/false 判断；

如：不使用 `if ( $a )` 或者 `if ( checka() )` 而使用 `if ( FALSE != $a )` 或者 `if ( FALSE != check() )`。

9、避免嵌入赋值

在程序中避免下面例子中的嵌入式赋值：

不使用这样的方式：

```
while ( $a != ( $c = getchar() ) ) {
    process the character
}
```

10、习惯与约定

通常变量的命名应该是有意义的单词，但在循环体中临时变量采用“IN规则”。

IN规则原本来自FORTRAN，在FORTRAN中，以字母表中I~N范围内字母开头的变量默认为整型变量。循环体中一般是整型变量，故习惯用I~N字母作为循环体中的变量命名。同时，I是标识符（Identify）首字母。如下：

```
function bubble_sort($array){
            $count = count($array);
            for($i=0;$i<$count; $s++){
              for($j=$count-1;$j<$i;$j--){
               if($array[$j]<$array[$j-1]){
                   $tmp = $array[$j];
                   $array[$j] = $array[$j-1];
                   $array[$j-1] = $tmp;
               }
              }
            }
            return $array;
}
```

编码规范

> 参考：ThinkPHP开发规范

1。普通变量采用小写字母,多单词以“`_`”连接，例如：`$base_dir`、`$red_rose_price`等。

2。静态变量以“`s_`”开头，字母小写，多单词以“`_`”连接，例如：`$s_base_dir`、`$s_red_rose_prise`等。

3。局部变量以“`_`”开头，字母小写，多单词以“`_`”连接，例如：`$_base_dir`、`$_red_rose_price`等。

4。全局变量以“ G_ ”开头，字母大写，多单词以“ _ ”连接，知道一个变量的作用域是非常重要的。例如： global $G_LOG_LEVEL 、 global $G_LOG_PATH 。

5。session以“ S_ ”开头，字母大写，多单词以“ _ ”连接，例子： $S_BASE_DIR 、 $S_RED_ROSE_PRICE 等。

6。确保文件的命名和调用大小写一致，是由于在类Unix系统上面，对大小写是敏感的（而ThinkPHP在调试模式下面，即使在Windows平台也会严格检查大小写）；

7。 类的声明按照“帕斯卡命名法”，例如： class MyClass 或 class DbOracle 等。

8。类文件要求以 .class.php 为后缀，使用驼峰法命名，并且首字母大写，例如 DbMysql.class.php 。

9。通常下划线开头的方法属于私有方法： _parseType 。

10。方法的命名使用“驼峰法”，或者使用下划线“_”连接，例如 getUserName 。

11。函数的命名使用小写字母和下划线的方式，例如 get_client_ip 。

12。属性的命名使用驼峰法，并且首字母小写或者使用下划线“ _ ”，例如 tableName 、 _instance ，通常下划线开头的属性属于私有属性；
以双下划线“ __ ”打头的函数或方法作为魔法方法，例如 __call 和 __autoload 。

13。常量以大写字母和下划线命名，例如 HAS_ONE 和 MANY_TO_MANY 。

14。类的命名空间地址和所在的路径地址一致，例如 Home\Controller\UserController 类所在的路径应该是 Application/Home/Controller/UserController.class.php 。

15。配置参数以大写字母和下划线命名，例如 HTML_CACHE_ON 。

16。语言变量以大写字母和下划线命名，例如 MY_LANG 。以下划线打头的语言变量通常用于系统语言变量，例如 _CLASS_NOT_EXIST_ ，
没有强制的规范，可以根据团队规范来进行。

17。类名和文件名一致（包括上面说的大小写一致）,程序中所有的类名唯一，例如 UserController 类的文件命名是 UserController.class.php ， InfoModel 类的文件名是 InfoModel.class.php ， 并且不同的类库的类命名有一定的规范。

```
//类统一采用：DemoTest.class.php
//接口统一采用：DemoTest.interface.php
//其他按照各自的方式：demoTest.{style}.php
```

18。数据表和字段采用小写加下划线方式命名，并注意字段名不要以下划线开头，例如 think_user 表和 user_name字段是正确写法，类似 _username 这样的数据表字段可能会被过滤。

19。确保文件的命名和调用大小写一致，是由于在类Unix系统上面，对大小写是敏感的；

20。PHP关键词及系统函数全部小写，`false/true/null` 也全部小写。

21。所有的PHP程序代码块标记均使用`<?php`，（短标签容易和xml混淆，php从5.4开始默认不支持短标记）。

22。对于只有php的代码文件，建议省略结尾处的‘`?>`’。这是为了防止多余的空格或其他字符影响到代码。

23、抽象类应以Abstract开头。

24。接口命名规则：
i）采用和类相同的命名规则，但在其命名前加‘`i`’字符，表示接口。如:iDataBase。
ii）尽量保持和实现它的类名一致。
接口文件可使用“iDatabase.interface.php”命名。

25、程序中所有类名唯一。

数据库命名规范

数据库表名

数据库表名命名遵循以下规范：

a．表名均使用小写字母；
b．表名使用统一前缀，且前缀不能为空（模块化，且可有效规避MySQL保留字）
c．对于多个单词组成的表名，使用_间隔；
d．对于普通数据表，使用 _t 结尾；
e．对于视图，使用 _v 结尾；
f．存储过程以 _proc 结尾；
j．触发器以 _tri 结尾；
h．Event调度以 _event 结尾。

例子：`user_info_t` 和 `book_store_v` 等。

数据库字段

数据库字段命名遵循以下规范：

a．全部使用小写；
b．多个单词间使用_间隔。
c．避免使用关键字和保留字。

例子：`user_name`、`rose_price` 等。

注释规范

每个程序均必须提供必要的注释，书写注释要求规范，参照PEAR提供的注释要求，为今后利用 phpdoc 生成 PHP 文档做准备。

1、程序头注释块

每个程序头部必须有统一的注释块，规则如下：

a．必须包含本程序的描述；

b．必须包含作者；

c．必须包含书写日期；

d．必须包含版本信息；

e．必须包含项目名称；

f．必须包含文件的名称；

g．重要的使用说明，如类的调用方法、注意事项等；

参考例子如下：

```
//
// +----------------------------------------------------------+
// | PHP version 4.0
// +----------------------------------------------------------+
// | Copyright (c) 1997-2001 The PHP Group
// +----------------------------------------------------------+
// | This source file is subject to of the PHP license,
// | that is bundled with this packafile LICENSE, and is
// | available at through the world-web at
// | http://www.php.net/license/2_02.txt.
// | If you did not receive a copy of the and are unable to
// | obtain it through the world-wide-web,end a note to
// | license@php.net so we can mail you a immediately.
// +----------------------------------------------------------+
// | Authors: Stig Bakken
// | Tomas V.V.Cox
//
// +————————————————————————————+
//
// $Id: Common.php,v 1.8.2.3 2001/11/13 01:26:48 ssb Exp $
```

2、类的注释

类的注释采用里面的参考例子方式：

```
/**
 * @ Purpose:
 * 访问数据库的类，以ODBC作为通用访问接口
 * @Package Name: Database
 * @Author: Forrest Gump gump@crtvu.edu.cn
 * @Modifications:
 * No20020523-100:
 * odbc_fetch_into()参数位置第二和第三个位置调换
 * John Johnson John@crtvu.edu.cn
 * @See: (参照)
 */
class Database {
    ...
}
```

3、函数和方法的注释

函数和方法的注释写在函数和方法的前面，采用类似下面例子的规则：

```
/**
 * @Purpose:
 * 执行一次查询
 * @Method Name: query()
 * @Param: string $queryStr SQL查询字符串
 * @Param: string $username 用户名
 * @Access: public
 * @Return: mixed 查询返回值（结果集对象）
 */
public function query （ $queryStr, $username ）{
    ...
}
```

4、变量或者语句注释

程序中变量或者语句的注释遵循以下原则：

a．写在变量或者语句的前面一行，而不写在同行或者后面；

b．注释采用 `/* */` 的方式；

c．每个函数前面要包含一个注释块。内容包括函数功能简述，输入/输出参数，预期的返回值，出错代码定义；

d．注释完整规范；

e．把已经注释掉的代码删除，或者注明这些已经注释掉的代码仍然保留在源码中的特殊原因。

例子：

```
/**
 * @Purpose:
 * 数据库连接用户名
 * @Attribute/Variable Name: db_user_name
 * @Type: string
 */
var db_user_name;
```

5、标注使用

IDE支持一些特殊注释，可以列出整个项目的特殊注释，方便后期的维护和代码检查，例如：

```
//@fixMe 表示需要修复项。如：修复了IP获取的一个安全漏洞
//@todo 表示需要完善的地方。如：这个函数的效率太低，需要改进。
```

上述注释和java中的标注及注解比较相像。可以查看NetBeans中标注的效果。

不同的IDE对这类特殊注释的支持程度不一。为了编码效率和团队协作，建议在项目开发时使用IDE进行项目管理。

对于代码不推荐使用的函数或方法，使用@Deprecated注释；对于重载方法，使用@over load注释。

其他规范

包含文件

提取出来具有通用函数的包含文件，文件后缀以 `.inc` 来命名，表明这是一个包含文件。

如果有多个 `.inc` 文件需要包含多页面，请把所有 `.inc` 文件封装在一个文件里面，具体到页面只需要包换一个 `.inc` 文件就可以了。

如： `xxx_session.inc` 、 `xxx_comm.inc` 、 `xxx_setting.inc` 、 `myssql_db.inc` 。

把以上文件以以下方式，封装在 `xxx.basic.inc` 文件里面：

```
require_once('xxx_session.inc');
require_once('xxx_comm.inc');
require_once('xxx_setting.inc');
require_once('mysql_db.inc');
```

PHP和HTML代码的分离问题

对性能要求不是很高的项目和应用，我们建议不采用 PHP 和 HTML 代码直接混排的方式书写代码，而采用 PHP 和 HTML 代码分离的方式，即采用模版的方式处理，这样一方面对程序逻辑结构更加清

晰有利，也有助于开发过程中人员的分工安排，同时还对日后项目的页面升级该版提供更多便利。

对于一些特殊情况，比如对性能要求很高的应用，可以不采用模版方式。

类的构造函数

如果要在类里面编写构造函数，必须遵循以下规则：

a．不能在构造函数中有太多实际操作，顶多用来初始化一些值和变量；

b．不能在构造函数中因为使用操作而返回false或者错误，因为在声明和实例化一个对象的时候，是不能返回错误的；

错误返回检测规则

检查所有的系统调用的错误信息，除非你要忽略错误。
为每条系统错误消息定义好系统错误文本，并记录错误LOG。

引用的使用

引用在程序中使用比较多，为了公用同一个内存，而不需要另外进行复制，XXX环境下的引用使用时，需要注意下面的情况；

在对函数的输入参数中使用引用时，不能在调用的时候在输入参数前加 `&` 来引用，而直接使用该变量即可，同时必须在函数定义的时候说明输入参数来自引用，比如下面的代码：

```
$a = 1 ;
function ab( &$var ) {
   $var ++ ;
   return $var ;
}

$b = ab($a); // 注意，此处不能使用 $b = ab(&$a)的方式；
echo $b."\n" ;
echo $a."\n" ;
```

此时 `$a` 和 `$b` 都是 2；
XXX环境下对引用的特殊要求源自 php.ini 文件里面的 allow_call_time_pass_reference 项设置，对外公开的版本是 `On` ，这样就可以支持 `&` 直接加到调用函数时变量前面进行引用，但是这一方法遭到抗议，并可能在将来版本的 PHP/Zend 里不再支持。受到鼓励的指定哪些参数按引用传递的方法是在函数声明里。你被鼓励尝试关闭这一选项（使用 `off` ，XXX的所有运行环境下都是 `off` ）并确认你的脚本仍能正常工作，以保证在将来版本的语言里它们仍能工作。

PHP项目开发中的程序逻辑结构

对于 PHP 项目开发，尽量采用 `OOP` 的思想开发，尤其在 PHP5 以后，对于面向对象的开发功

能大大提高。

在 PHP 项目中，我们建议将独立的功能模块尽量写成函数调用，对应一整块业务逻辑，我们建议封装成类，既可以提高代码可读性，也可以提高代码重用性。比如，我们通常将对数据库的接口封装成数据库类，有利于平台的移植。

重复的代码要做成公共的库。（除了我们在 `plug-in` 产品上遇到的情况，该产品系列有多个相类似的产品，为了尽可能地减少安装包尺寸，不适合将这些产品共用的所有函数做成公共的库）。

PHP大小写敏感问题整理

> 参考：http://www.jbxue.com/article/9280.html

php的大小写敏感问题整理：

我们知道，PHP语言对大小写敏感问题的处理有点乱，在编程时需要多加注意，特别是在linux平台中更要注意。
说明：
不鼓励用这些规则，仅作为学习研究之用。

> 在编程中，应当坚持“*大小写敏感*”，遵循统一的代码规范。

一、大小写敏感

1. 所有变量均区分大小写，包括普通变量以及PHP 超级全局变量。

```
<?php
$abc = 'abcd';
echo $abc; //输出 'abcd'
echo $aBc; //无输出
echo $ABC; //无输出
?>
```

2 .常量名默认区分大小写，通常都写为大写。

```
<?php
define("ABC","Hello World");
echo ABC;   //输出 Hello World
echo abc;   //输出 abc
?>
```

3 .php.ini配置项指令区分大小写

如 file_uploads = 1 不能写成 File_uploads = 1。

二、大小写不敏感

1. 函数名、方法名、类名 不区分大小写，但推荐使用与定义时相同的名字。

```
<?php
function show(){
    echo "Hello World";
}
show(); //输出 Hello World    推荐写法
SHOW(); //输出 Hello World
?>
```

```
<?php
class cls{
     static function func(){
        echo "hello world";
     }
}
Cls::FunC(); //输出hello world
?>
```

2 .魔术常量不区分大小写，推荐大写,包括：LINE、FILE、DIR、FUNCTION、CLASS、METHOD、NAMESPACE。

```
<?php
echo __line__; //输出 2
echo __LINE__; //输出 3
?>
```

3 .NULL、TRUE、FALSE不区分大小写。

```
<?php
$a = null;
$b = NULL;
$c = true;
$d = TRUE;
$e = false;
$f = FALSE;
var_dump($a == $b); //输出 boolean true
var_dump($c == $d); //输出 boolean true
var_dump($e == $f); //输出 boolean true
?>
```

4 .类型强制转换，不区分大小写。

```
<?php
$a=1;
var_dump($a); //输出 int 1
$b=(STRING)$a;
var_dump($b); //输出string '1' (length=1)
$c=(string)$a;
var_dump($c); //输出string '1' (length=1)
?>
```

SQL注意事项

1、SQL语句不区分大小写(要大写全大写,小写全小写),但数据库和表的名称则区分大小写!
2、 `NOT NULL` 的意思是表中所有行的此属性必须有一个值;如果没有指定,该列可以为空(`NULL`);
3、 `unsigned` 无符号的,意思是它只能是0或者一个正数;

PHP与JS运算符优先级比较

PHP与JS的基本语法结构非常相似，即便是在职多年的老手特别是双管前端与后台开发的程序员，常常会搞错某些最基础的东西。下面将PHP与JS的运算符列出，以便集中巩固基础知识。

PHP运算符优先级

方向	运算符	附加信息
非	clone new	clone 和 new
左	[	array()
非	++ --	递增 / 递减运算符
非	~ - (int) (float) (string) (array) (object) (bool) @	类型
非	instanceof	类型
右	!	逻辑操作符
左	* / %	算术运算符
左	+ - .	算术运算符和字符串运算符
左	<< >>	位运算符
非	< <= > >= <>	比较运算符
非	== != === !==	比较运算符
左	&	位运算符和引用
左	^	位运算符
左	|	位运算符
左	&&	逻辑与运算符
左	||	逻辑或运算符
左	? :	三元运算符
右	= += -= *= /= .= %= &= |= ^= <<= >>=	赋值运算符
左	and	逻辑运算符
左	xor	逻辑运算符
左	or	逻辑运算符
左	,	多处用到

JS运算符优先级

方向	. [] ()	字段访问、数组下标、函数调用以及表达式分组
非	++ - - ~ ! delete new typeof void	一元运算符、返回数据类型、对象创建、未定义值
左	* / %	乘法、除法、取模
左	+ - +	加法、减法、字符串连接
左	<< >> >>>	移位
左	< <= > >= instanceof	小于、小于等于、大于、大于等于、instanceof
非	== != === !==	等于、不等于、严格相等、非严格相等
左	&	按位与
左	^	按位异或
左	|	按位或
左	&&	逻辑与
左	||	逻辑或
左	?:	条件
右	= += -= *= /= .= %= &= |= ^= <<= >>=	赋值、运算赋值
左	,	多重求值

运算符的方向中，左表示同级的（不必同一种运算符）表达式从左向右求值，右表示表达式从右向左求值。

编写高效的CSS

编写好的CSS代码，有助提升页面的渲染速度。本质上，引擎需要解析的CSS规则越少，性能越好。MDN上将CSS选择符归类成四个主要类别，如下所示，性能依次降低。
ID 规则 Class 规则 标签规则 通用规则 对效率的普遍认识是从Steve Souders在2009年出版的《高性能网站建设进阶指南》开始，虽然该书中罗列的更加详细，但你也可以在这里查看完整的引用列表，也可以在谷歌的《高效CSS选择器的最佳实践》中查看更多的细节。

本文我想分享一些我在编写高性能CSS中用到的简单例子和指南。这些都是受到MDN 编写的高效CSS指南的启发，并遵循类似的格式。

显然，这里只讲述了少数的规则，是我在我自己的CSS中，本着更高效和更易维护性而尝试遵循的规则。如果你想阅读更多的知识，我建议阅读MDN上的编写高效的CSS和谷歌的优化浏览器渲染指南。

1、遵循一个标准的声明顺序

虽然有一些排列CSS属性顺序常见的方式，下面是我遵循的一种流行方式。

```
.someclass {
    1.位置属性(position, top, right, z-index, display, float等)
    2.大小(width, height, padding, margin)
    3.文字系列(font, line-height, letter-spacing, color- text-align等)
    4.背景(background, border等)
    5.其他(animation, transition等)
}
```

box-model从内向外写： size -> padding -> border -> margin

后代选择符最烂

不仅性能低下而且代码很脆弱，html代码和css代码严重耦合，html代码结构发生变化时，CSS也得修改，这是多么糟糕，特别是在大公司里，写html和css的往往不是同一个人。

```
// 烂透了
html div tr td {..}
```

避免链式（交集）选择符

这和过度约束的情况类似，更明智的做法是简单的创建一个新的CSS类选择符。

```
// 糟糕
.menu.left.icon {..}

// 好的
.menu-left-icon {..}
```

组织好的代码格式

代码的易读性和易维护性成正比。下面是我遵循的格式化方法。

```
// 糟糕
.someclass-a, .someclass-b, .someclass-c, .someclass-d {
 ...
}

// 好的
.someclass-a,
.someclass-b,
.someclass-c,
.someclass-d {
 ...
}

// 好的做法
.someclass {
    background-image:
        linear-gradient(#000, #ccc),
        linear-gradient(#ccc, #ddd);
    box-shadow:
        2px 2px 2px #000,
        1px 4px 1px 1px #ddd inset;
}
```

Web前端开发流程

前端webpack workflow

gulp

bower

sass是一种对css的一种提升，可以通过编译生成浏览器能识别的css文件。sass技术的文件的后缀名有两种形式：.sass和.scss。
http://javaloverlover.iteye.com/blog/1603314

Yeoman是由Paul Irish、Addy Osmani、Sindre Sorhus、Mickael Daniel、Eric Bidelman和Yeoman社区共同开发的一个项目。它旨在为开发者提供一系列健壮的工具、程序库和工作流，帮助他们快速构建出漂亮、引人注目的Web应用。Yeoman拥有如下特性：

> 1、快速创建骨架应用程序——使用可自定义的模板（例如：HTML5、Boilerplate、Twitter Bootstrap等）、AMD（通过RequireJS）以及其他工具轻松地创建新项目的骨架。
> 2、自动编译CoffeeScrip和Compass——在做出变更的时候，Yeoman的LiveReload监视进程会自动编译源文件，并刷新浏览器，而不需要你手动执行。
> 3、自动完善你的脚本——所有脚本都会自动针对jshint（软件开发中的静态代码分析工具，用于检查JavaScript源代码是否符合编码规范）运行，从而确保它们遵循语言的最佳实践。
> 4、内建的预览服务器——你不需要启动自己的HTTP服务器。内建的服务器用一条命令就可以启动。
> 5、非常棒的图像优化——Yeoman使用OptPNG和JPEGTran对所有图像做了优化，从而你的用户可以花费更少时间下载资源，有更多时间来使用你的应用程序。
> 6、生成AppCache清单——Yeoman会为你生成应用程序缓存的清单，你只需要构建项目就好。
> 7、"杀手级" 的构建过程——你所做的工作不仅被精简到最少，让你更加专注，而且Yeoman还会优化所有图像文件和HTML文件、编译你的CoffeeScript和Compass文件、生成应用程序的缓存清单，如果你使用AMD，那么它还会通过r.js来传递这些模块。这会为你节省大量工作。
> 8、集成的包管理——Yeoman让你可以通过命令行（例如，yeoman搜索查询）轻松地查找新的包，安装并保持更新，而不需要你打开浏览器。
> 9、对ES6模块语法的支持——你可以使用最新的ECMAScript 6模块语法来编写模块。这还是一种实验性的特性，它会被转换成eS5，从而你可以在所有流行的浏览器中使用编写的代码。
> 10、PhantomJS单元测试——你可以通过PhantomJS轻松地运行单元测试。当你创建新的应用程序的时候，它还会为你自动创建测试内容的骨架。

前端开发环境搭建 GRUNT BOWER、REQUIREJS 、 ANGULAR