**一、引言**

本文的核心观点为CSS的合并与模块化，似乎与前一篇文章“[CSS样式的再分离](http://www.zhangxinxu.com/wordpress/?p=916)”有矛盾，其实不然，分离可以精简CSS代码，合并也可以精简CSS代码，**一切都是权衡！**或是说是在恰当的情况下使用恰当的手段。

正如前文所提到的，分离可以精简CSS，但是同时会带来巨大的HTML代码的开销，显然，对所有的样式进行再分离式不切实际的，是会带来痛苦的。前文提到的“通用库”看似属于分离，其实又是分离之外的东西。“通用库”属于很良性的东西，任何网站都可以拿来用，不会产生什么副作用，因为其与当前项目的结构，样式表现没有必然的关联。“通用库”就像是一个公共资源，大家都可以来采撷。但是“实际项目库”却是个烫手的山芋，这是根据当前实际项目分离出来的独立样式集合，我们可能会分理出页面中常见的背景色样式（如background-color:#f7f7f7;），可那会分理出特定的粗边框样式（如border:3px solid #c80000;），一般情况下，这是很ok的，但是，如果一些模块化的样式（例如整站通用的按钮）也是使用的分离样式组合而成，那么，后期要是修改按钮样式，就会很痛苦，因为会有那么多的样式要替换。

所以，盲目的分离是会带来恶果的。

本文的“合并”和“分离”属于对立又相辅相成的，理解的“合并”与“分离”的精髓之后，您会发现写CSS代码就是一门艺术。同样的，本文也是为我后面的“我的CSS架构”一文做铺垫的，写这些都是为了同一个目的：写出最精简高效的CSS代码。

**二、明确“模块化”专指“页面元素的模块化”**

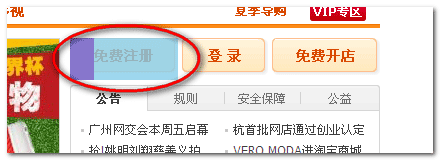
首先您要明确，样式再分离是应用到“模块化的独立元素”上可那会产生后期维护的问题，并不是应用到“页面模块”会产生后期维护的问题。例如，我们将很多分离的样式嵌入到一个整站通用的的“评论模块”中，是不会产生任何所谓的后期维护的问题的，除非您网站的评论并不是个“模块”，而是这里一段评论的HTML代码，那里又是另外一评论的HTML代码，有经验的开发人员都应该清楚我想要表达的意思。

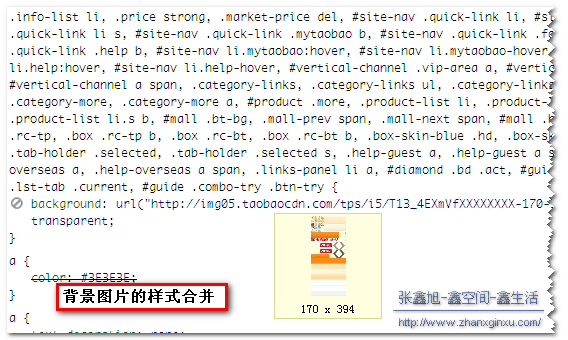
本文标题所说的“模块化”指的是页面元素，例如网站通用按钮，通用选项卡，通用小图标，或是页面的一些固定框架结构等。这些元素是不适宜使用样式再分离的（或者说仅仅使用样式再分离）。

**三、什么是CSS样式合并**

何为CSS样式合并，所谓CSS样式合并，指的是一些不可分离的样式（按钮，图标等），将他们公共的样式部分进行合并，非公共的再次独立出来，以减小CSS文件的大小。我想，合并的做法很多同行都做过，可能不是很彻底，或是系统。很多时候，我们知道合并的好处，但是往往由于各种原因，没有从整体对样式进行设计与架构，造成样式合并的效果基本上没有发挥出来。下面我举个实例，会让您对样式合并有一个更进一步的认识。此实例来自淘宝首页，其对背景图片的样式合并。

zxx://淘宝首页的Flash logo很有爱啊，观看点击[这里](http://img03.taobaocdn.com/tps/i3/T1TudFXo8GXXXXXXXX.swf" \t "_blank)。

我们使用小bug（我对firebug的昵称）随便看一个带背景图片的元素，例如下面这个（免费注册按钮）：  


此时firebug右侧显示的内容截图如下：  


其对所有使用到这张sprite背景图的样式进行了合并，试想下，淘宝的背景图片地址这么长，加入这些样式不合并，那么首页的CSS大小增加的量可能要上K了，对于淘宝首页这样大流量的的页面来说，增加1K的大小，就是要从马云手中拿走成百上万的money~~

就我自己而言，使用最多的合并也是背景图片的合并。其次就是一些效果类似但又不完全一致的模块化元素。样式的合并，没有规律性可言，一般，遇到结构或是写法类似但又不完全一样的元素的时候，就可以使用样式的合并。

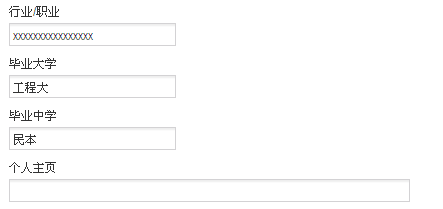
使用英文字符的逗号(,)分隔样式名，将相同的样式写在后面，这也些类似于初中数学里的“合并同类项”。项目不同，情况也各异，要想达到充分的样式合并，前期的设计与架构很重要。

**四、CSS分离与CSS合并的共存**

CSS“通用库”游离于三界之外，不参与这类纷争（例如与其他元素合并）。这里的CSS分离指的就是在实际项目中分离出来的“实际项目库”。一般情况下，“分离”与“合并”处于CSS文件的不同部分，两者是不搭噶的。“分离”一般针对那些非模块化的元素，而“合并”多针对模块化的元素，所以两者是对立的属于不同类别的，之间不会产生什么冲突。由于两者都有精简CSS代码的作用，所以双管齐下，事半功倍。

虽说“井水不犯河水，鸡腿有别鸭腿”，但是河水泛滥，家禽玩蛋之时，两者也会产生交集的。“分离”与“合并”也是如此。这不是一句话能够说清楚的，带我娓娓道来。

前面提到，模块化元素是不适宜使用分离的。比说如，文本框，设计师们往往喜欢在文本框上打主意，例如添加个淡灰渐变背景什么的，例如下面的效果（为截图）：



这里的文本框就是整站通用的独立的“模块化元素”，是不推荐使用分离的。总共整个网站，文本框的宽度有好几种，从宽度50像素左右的，200像素左右的，到450像素左右的都会有，我们不可能针对每个宽度写一个独立的样式的。显然，这里需要对文本框样式进行合并，将公共的部分独立出来，于是，我们可能会有如下的代码（其中inset的背景与其他背景图片元素进行合并了，所以这里只有background-position属性）：

.inset{

height:16px;

background-position:0 -220px;

background-color:white;

border:1px solid #D3D2D4;

padding:3px 0 2px 2px;

}

好的，以上就是inset的公共样式，看上去像是分离，其实是合并。现在唯一落下的就是宽度属性了，这里宽度属性为单一的属性，我们是不是可以用“样式分离”的意识将其分离出来，与其他元素公用呢，就像下面：

.w163{width:163px;}

.w297{width:297px;}

.w397{width:397px;}

.w710{width:710px;}

然后使用input时，HTML代码如下：

宽度1：<input type="text" class="inset w163" />

宽度2：<input type="text" class="inset w297" />

宽度3：<input type="text" class="inset w397" />

同时，分离出来的w163又能被页面其他地方的元素重用，岂不是很好。好吗？一点都不好！记住，模块化的元素千万不能用“分离”的思想来处理！我们试想一下，要是项目全部完成了，主管说：“这个文本框宽度有点长，你给我改短一点”，试问，您要怎么改？w163本身就已经分离并语义明确化，就算你class名不变，修改后面的163px为140px，确实，所有的应用w163样式的文本框宽度减小了，但是，项目上其他应用了w163样式的元素就遭殃了，这就是我提到的“痛苦”。避免以后产生痛苦，一定要技术这里原则：**模块化元素不分离**！

所以，这里的意识不能停在“分离”上，“分离”还是“不分离”主要体现在命名上，“分离”思想下的命名只针对属性本身，犹如机器般，是没有情感的，“无分离”思想的命名是针对页面内容，命名可以反映其所知道的大体内容，有血有肉的，所以，上述文本框的宽度可以使用类似下面的命名方式：

.txtw1{width:163px;}

.txtw2{width:297px;}

.txtw3{width:397px;}

.txtw4{width:710px;}

使用使应该如下html代码：

宽度1：<input type="text" class="inset txtw1" />

宽度2：<input type="text" class="inset txtw2" />

宽度3：<input type="text" class="inset txtw3" />

虽然txtw1也是分离出来的单样式，但是由于其命名带有内容语义，所以，不会被贸然当作分离样式在其他地方适应，而带来后期维护潜在的问题。

还没完，其实上面都是讲的“非分离”，不是讲“共存”，似乎有点不切题。不急，下面才是关键。这里，txtw1明明是独立样式，却不能当作分离样式使用，这种心情就像是看到花季少女跳楼般——可惜了！实则不然，这里txtw1的样式是可以当作“分离”样式使用的，如何使用？关键就是本节的关键字“合并”。我们可以将分离样式与内容语义的独立样式进行合并，就可以实现样式分离重用又没有后期维护潜在风险的问题。比方说，我在做一个列表，这个列表的宽度就是163像素，嘿，正好，CSS中有个宽度为163像素的独立样式txtw1，但是这个到嘴的肥鸭却吃不得（前面提到的潜在风险），怎么办，我们可以分离出一个163像素宽度的样式w163，同时与txtw1进行合并，这样，就实现了样式重用，又规避了潜在风险，于是，我们会有如下的样式代码：

**.txtw1,.w163{width:163px;}**

.txtw2{width:297px;}

.txtw3{width:397px;}

.txtw4{width:710px;}

这就是“分离”与“合并”共存的极佳实例。

要注意，只有页面要使用到与文本框独立宽度一样宽度的时候才进行分离并合并，所以以下做法是属于犯傻不可取的：

.txtw1,.w163{width:163px;}

.txtw2{width:297px;}

.txtw3{width:397px;}

.txtw4{width:710px;}

现在，我们要修改txtw1样式下的文本框宽度为150像素，您直接改就行了，只有文本框应用了这个样式，不会有潜在问题。但是，记得，这里的合并要还原，如下：

.txtw1{width:150px;}

.w163{width:163px;}

可见，真正理解了“合并”与“分离”，就不会出现什么维护上的风险，相反，可以最大限度的发挥两者共有的精简CSS的特性，相辅相成。

**五、精简高效的CSS代码是通力合作的结果**

就算你CSS再牛叉，理解再深刻，没有设计师以及后台程序工程师的配合，您的CSS代码会不得不变得不高效，甚至难以维护。

关于设计师的作用，我在[前文](http://www.zhangxinxu.com/wordpress/?p=916)已经提到，这里不再多说。关于后台程序员，这里简单说下。样式再分离，最怕的就是整站通用的模块化元素。如果网站系统架构良好，凡事功能一致的模块都是仅仅一段代码，或是一个页面片段，整站通用的，此时，“样式再分离”的优势和潜力可谓发挥到了极致，分离样式可以很放心的重用，很放心的分离（通用单元素和复杂结构除外），因为即使后期修改，也只会改一个地方，分离所带来的多点维护的问题就无从谈起，分离只有百益而无一弊。但是，要是碰到“散枪法”的后台程序员，没有模块化的思想，系统没有良好的架构，类似功能遍地开花，啊哦！my lady gaga！我实在不敢想像此时应用样式分离那种饿殍遍野的惨况。

波音747的马达装载拖拉机上是不顶鸟用的。“分离”+“合并”的高效CSS代码是需要设计师，前端开发工程师，后台开发工程师都很专业的情况下才能发挥巨大效用的，如果不是这种情况，我建议还是沿着主流CSS的路子走，否则会有苦头吃的。

**六、结语**

一些个人观点，可能与您认识相左，欢迎交流。要是有您认为表述不准确的地方，也欢迎指正。  
就这些。