**从模块化CSS到面向对象的CSS**

最近做了一个项目，尝试了一些以前未曾上尝试过的方法——基于模块设计整个网站的内容。做到后期才发觉如果初期的时候设计的更好一些，能做到面向对象的CSS来设计，那整体开发效率以及代码复用方面会做比现在好。现在的代码还是有点不干净的地方，有些地方有点乱。这可能是这行当的“完美主义”情结吧 :) 希望下次可以做的更好。

### 1.模块化问题

首先，什么是模块化？我这里指的模块化并非仅仅指CSS模块化，还指的是HTML的模块化。那为什么要模块化？最主要特点是使各个部分相对独立，同时增加代码复用，最终目的是为了降低开发成本提升开发效率。

举个例子，当拿到一个项目的时候，扫一遍所有设计，就可以发现：  
哦，一共有10种侧边栏设计，设计师就是把这10种侧边栏翻来覆去放在不同的页面中。那就很简单了，只要完成了这10种侧边栏，那么就算100个1000个页面也都是一样的，只要把HTML复制，粘贴就可以了。最大化的实现的代码复用。

鬼哥写过一篇文章来阐述这个问题：  
<http://www.cssforest.org/blog/index.php?id=134>

### 2.面向对象的CSS

当进一步观察，很容易发现，有侧边栏都是有一个标题，再下面是是主面板，再有一类侧边栏是标签页的，下面展示的是新闻之类的内容。这个时候就完全可以抽象出两个类，一个是带标题的侧边栏（title\_box），另一个是带标签的侧边栏(tab\_box)。这样的好处是可以进一步抽象出代码，提高复用率。增加开发速度。  
比如前者的HTML可以是这样的：

[更多](http://shawphy.com/2009/08/object-oriented-css.html)

### 标题

* [新闻](http://shawphy.com/2009/08/object-oriented-css.html)
* [新闻](http://shawphy.com/2009/08/object-oriented-css.html)

可能的CSS是：  
  
.title\_box {margin-bottom:10px;border:1px solid #aaa;}  
.title\_box h3{border-bottom:1px solid #aaa;}  
.title\_box .act{float:right}  
.title\_box .panel{margin:0 10px;}  
  
为了保证代码的简洁易读，一些小的padding或者背景修饰我就不写了。  
其中act是action的意思，这里[偷学](http://lifesinger.org/blog/2008/10/youa-f2e-tech-review/)来的XD

是的，以上的HTML以及CSS代码就共同构建了一个基本的侧边栏类  
他所具有的特诊是有边框，下面空开10px，右上角还有个“更多”

有同学可能会问，我为何要把act类放在那个span上而不是a上？为何要把panel放在一个div上？放在ul上不是可以减少代码么？这里我之所以这样写，是为了最少代码与开发成本之间平衡的考虑。所有成本中，人力成本是最高的，服务器有钱都可以加，但是代码得更快速的写出来才是要紧的。这么写的话可以最大保证通用性。同时，另一个理论用于支持这种写法的是：只有无语义的div和span适合拿来做布局，而有语义的元素不应当参与到布局中。当然，我这里只是一种权衡。

这么写是很灵活的一种写法。如果不需要“更多”，直接删除即可，如果不需要标题，直接删除即可。进一步来看，这段HTML可以应用到整个网站各个地方而不仅限于侧边栏。只要是形式相同的地方都可以用。

### 3.进一步实现继承和扩展

这里带来的一个问题是如何控制宽度。侧边栏和正文部分宽度不同，有些地方需求又各异，如何保证宽度？跳出思维局限来看，是不是所有某一宽度的div都同属一类？是的，他们是同一类的，所以只要定好了宽度，继承就可以了！

如果应用了栅格化布局，就能很容易理解这一点，那么上面的代码就应该是这样的

前面的g5就代表了其占了5个栅格！这个模块首先是占了5个格子的宽度，然后他是一个带标题的模块。进一步看，如果这个格子在不同的地方背景色不同，那可以继承了前面两个类的基础上，再加上一个 bg\_blue于是这就成了：

没错，他就是一个占5个格子的带标题的模块，背景色是蓝色的。  
好，可能他在某个地方是左浮动的，需要加上float:left，但不是所有的title\_box都需要做浮动，那自然加上一个left类即可：

所以，挑选一个合适的基于栅格的框架很有意义，如YUI,BluePrint，或者OOCSS。  
可以参考一下OOCSS，我很欣赏其中的思想  
前端观察对其的介绍：<http://www.qianduan.net/object-oriented-css.html>

针对面向对象CSS的一个很好的PPT  
<http://www.slideshare.net/stubbornella/object-oriented-css>

面向对象的CSS框架（OOCSS）的项目地址  
<http://wiki.github.com/stubbornella/oocss>

我除了不喜欢其中到处都是为了圆角而设计的CSS之外，其他都很喜欢。

另外，写的时候，我搜索了一下，才发现这种思想08年初嗷嗷就已经有啦，我Out啦：  
<http://www.aoao.org.cn/blog/2008/01/oo-style/>

### 4.这不是银弹

要坚信，在解决一个问题的同时必然会带来其他问题。

**优先级问题**  
其中一个问题是优先级问题，由于模块细分，优先级很容易弄得焦头烂额。我的经验是每个类都负责一个部分——布局归布局，色彩背景归色彩背景。只有把各个类细分出来，才能避免样式的冲突。最后，id也是必不可少的。也需要important。比如上面，我就会定义一个noborder类，表示这个容器一定没有边框。用于个别地方是一个title\_box但没有边框的情况。这个类里我就设置了  
  
.noborder{border:0 !important;}  
  
这样就能保证在任何时候，都不会有边框了。

同时，我对于文件规划，通常是一个页面只引入这个页面的css，比如首页的index.css  
在这个css里再用import依次导入用于  
重置的reset.css（重置标签类，其实一个标签本身不就是一个类么？div类，span类，本身就具有block或者inline属性，我们只是在其上面在扩展而已）  
global.css（用于设置header和footer以及中间主体部分的布局，因为这两个部分一般整站是不会变的）  
layout.css（用于设置栅格，构建内容以及侧边栏的宽度等。）  
module.css（用于建立如title\_box这种中等模块）  
submod.css（这里放一些细节的模块，比如某一个新闻列表的完整细节样式设定）  
如果模块较多，还可能分成若干的子块，侧边栏放侧边栏，中间的放中间。因为我相信绝大多数情况下侧边栏将永远是侧边栏，而不会跑到中间去。  
如果有必要，也会建立form.css，专门设置表单元素，定义表单元素的外观。我会在里边设置一些 blue\_btn，red\_btn之类的class来定义不同颜色的button.  
而后，在这个首页的index.css里写特定的样式。  
这种规划的方法在很多文章中都能见到影子，如今再回头看，又有了更深一层的体会。

另外，import有潜在的性能问题。[见这里](http://www.qianduan.net/high-performance-web-site-do-not-use-import.html)。  
只是我实在抵御不住import带来的便利性，在html里加入一堆的link是多么痛苦的事情。而且大多数时候对性能的要求没那么高，为此我宁可节约一点人力成本。这是种取舍，因网站需求而异。但总体顺序还是不变的。  
而解决这个性能问题的方法是用 条件CSS，可以参考[这里](http://www.qianduan.net/conditions-for-the-use-of-css.html)  
和[这里](http://www.qianduan.net/conditions-of-high-level-usage-of-css.html)  
**id与class**  
命名的时候到底是以id为主，还是class为主，想必各位同学看到这里也该明白在下的观点了——以class为主，id为辅。因为同一段HTML可能用到页面的任何地方，而id在同页面里是不能重复的。而且，为了体现继承扩展的关系，应当使用class了。  
那么id还用不用？当然要用，且不论id在写JS时候的速度，实际在写页面的时候，肯定会某一个模块做“个性化”定义，这就需要使用id了。个性化的东西写在id里，这样就不用担心干扰其他模块了，最大避免触一发动全身的窘境。通用与独立同在。

聪明的同学一定想到了什么，对，是不是记得曾经看到过启蒙读物上，一再强调，千万不要把一个类明明为left，因为指不准哪天他就不在左边了。不要定义为blue，因为哪天就改成红色了？是的，那个说没错，但我这里也没错。我们说的是两个东西。

**修改CSS还是修改HTML**

曾经看到的那些，都是建立在一旦HTML写完，需求变的时候不许改HTML，只许改CSS的基础上的。但实际上，又多少机会能让你只改CSS而不改动HTML的呢？另外，既然有一套完善的CSS了，为何不直接简单修改一下HTML呢？如果背景变变成了红色，我只需要修改一下class，从bg\_blue改成bg\_red就可以了，何乐而不为？由此看来，class取名成left和right并非很糟糕的决定，只要合理利用，完全可以应付多样的需求。

**开发成本VS维护成本**  
这样做之后带来的问题就是开发成本与维护成本的矛盾。这样开发确实很快速。但问题是以前小改动起来是不需要动HTML的，而现在改一次就得改HTML，而这方面一旦页面做好后就可能归其他部门那边管了，修改测试上传都得惊动其他部门，这方面如何协调是一个问题。  
换句话说：面向对象CSS保持CSS框架不变而修改HTML，而原先那种是保持HTML不变而修改CSS  
这方面我没有太多经验，欢迎大家探讨。  
Twinsen有过类似的探讨( http://www.twinsenliang.net/skill/20081129.html ) 他就很反对这么来。  
其中的大部分观点都赞同，只是最近我也在反思：  
到底多大机会能让我碰上“left\_box”和“right\_box”的互换。全站的blue\_button换成红色呢？  
如果只是一两个页面的话，修改CSS的成本更低，还是保持CSS不动修改HTML的成本更低？

**增加学习成本**  
当公司招募新人的时候，不得不对其进行培训，以使其学会使用这套自建的框架体系。但带来的好处是使整个网站风格统一，给进一步维护带来方便。所以我认为增加的这点学习成本是值得的。  
同时，如果你都已经跳槽到别处了，没留下好的文档，人家学不会也没关系。直接抛开这套体系也能过日子。因为本身这套体系是从reset开始对各个标签初始化对象的，而后逐渐增加class以扩展对应属性。  
也就是说，新人也可以抛开这套东西，直接用自己的写法来写。当然，这好不好已经不在今天的讨论范围了。这里只是为了说：这个框架也可以让新手不管这些约束直接写自己的。

**class过长导致文件过大**  
这可以通过压缩来解决。用正则找出css和html中所有的class，以出现频率排序，然后分别从abcdeft…开始，先单字母，后双字母 a一直到zz应该有26\*27种组合，足够一般网站应用了。google应该也是这样做的。抱歉这段时间比较慢，代码就没写了。

副作用：js必须用id而不是class了，否则压缩后会找不到对应的class  
解决思路：约定JS中保存class的字符串变量为 foo\_cls=”bar”，那么只要用正则找到对应\_cls结尾的变量进行对应替换即可。  
这本身是面向对象的CSS，而进一步压缩这里仅提供一种思路。

### 总结

结构和样式分离是好事，我这里提出的观点反而有点逆潮流，试图通过HTML中的class属性来控制表现。我认为这是一种理性回归，至少我们不用一个ID下面写一堆的针对他的样式，冗余，容易出错，效率还高。同时这种方式与直接在元素上写style也是有本质区别的，模块化，面向对象的思想体现在其中，合理控制每一个类的颗粒度，或者说控制范围。  
我这种介于结构—样式的完全分离与结构—样式完全整合之间的一种状态，试图寻找一种平衡，提升开发效率并降低维护成本。

谨以此文做以探讨。

### 扩展阅读

[重构之美－跨越Web标准，触碰语义网[分离：通用也许是个美丽陷阱]](http://www.cnblogs.com/yuntian/archive/2008/09/23/1269800.html)——另一个反对的声音  
[面向对象的CSS－－OOCSS](http://www.cnblogs.com/hdjjun/archive/2008/10/15/1311532.html)国人写的通过JS增加CSS面向对象功能