CSS规范 - 优化方案

值缩写

缩写值可以减少CSS文件大小，并增加可读性和可维护性。

但并非所有的值都必须缩写，因为当一个属性的值缩写时，总是会将所有项都设置一遍，而有时候我们不希望设置值里的某些项。

```
/* 比如我们用下面这个样式来让某个定宽的容器水平居中，我们要的只是left和right，
 * 而top和bottom不是这个样式要关心的（如果设置了反倒会影响其他样式在这个容器上的使用），
 * 所以这时我们就不需要缩写
 */
.f-mgha{margin-left:auto;margin-right:auto;}
/* 比如下面这个模块的样式设置，我们确实需要设置padding的所有项，于是我们就可以采用缩写 */
.m-link{padding:6px 12px;}
常用的缩写方法请参见代码格式。
```

避免耗性能的属性

以下所举列的属性可能造成渲染性能问题。不过有时候需求大于一切……

```
/* expression */
.class{width:expression(this.width>100?'100px':'auto');}
/* filter */
.class{filter:alpha(opacity=50);}
```

选择器合并

即CSS选择器组合，可以一次定义多个选择器，为你节省很多字节和宝贵时间。

通常我们会将定义相同的或者有大部分属性值相同（确实是因为相关而相同）的一系列选择器组合到一起（采用逗号的方法）来统一定义。

```
/* 以下对布局类选择器统一做了清除浮动的操作 */
.g-hd:after,.g-bd:after,.g-
ft:after{display:block;visibility:hidden;clear:both;height:0;content:".";}
.g-hd,.g-bd,.g-ft{zoom:1;}
/* 通常background总是会占用很多字节，所以一般情况下，我们都会这样统一调用 */
.m-logo,.m-help,.m-list li,.u-tab li,.u-tab li
a{background:url(../images/sprite.png) no-repeat 9999px 9999px;}
.m-logo{background-position:0 0;}
/* 以下是某个元件的写法，因为确实很多元素是联动的或相关的，所以采用了组合写
法，可以方便理解和修改 */
.u-tab li,.u-tab li a{display:inline;float:left;height:30px;line-
height:30px;}
.u-tab li{margin:0 3px;}
.u-tab li a{padding:0 6px;}
```

背景图优化合并

图片本身的优化：

- 图像质量要求和图像文件大小决定你用什么格式的图片，用较小的图片文件呈现较好的图像质量。
- 当图片色彩过于丰富且无透明要求时，建议采用jpg格式并保存为较高质量。
- 当图片色彩过于丰富又有透明或半透明要求或阴影效果时，建议采用png24格式，并对IE6进行png8退化（或在不得已情况下使用滤镜）。
- 当图片色彩不太丰富时无论有无透明要求，请采用png8格式，大多数情况下建议采用这种格式。
- 当图片有动画时，只能使用gif格式。
- 你可以使用工具对图片进行再次压缩，但前提是不会影响色彩和透明。

多张图片的合并：

- 单个图标之间必须保留空隙，空隙大小由容器大小及显示方式决定。这样做的好处是既考虑了“容错性”又提高了图片的可维护性。
- 图标的排列方式，也由容器大小及显示方式决定。排列方式分为以下几种：横向排列（容器宽度有限）、纵向排列（容器高度有限）、斜线排列（容器宽高不限），靠左排列（容器背景居左）、靠右排列（容器背景居右）、水平居中排列（容器背景水平居中）、垂直居中排列（容器背景垂直居中）。
- 合并后图片大小不宜超过50K，建议大小在20K-50K之间。
- 为保证多次修改后的图片质量，请保留一份PSD原始图，修改和添加都在PSD中进行，最后导出png。

分类合并：

并不是把所有的图标都合并在一张图片里就是最好的，除了要控制图片大小之外还要注意以下方法。

- 按照图片排列方式，把排列方式一样的图片进行合并，便于样式控制。
- 按照模块或元件，把同属于一个模块或元件的图片进行合并，方便模块或元件的维护。
- 按照图片大小，把大小一致或差不多的图片进行合并，可充分利用图片空间。
- 按照图片色彩，把色彩一致或差不多的图片进行合并，保证合并后图片的色彩不过于丰富，可防止色彩失真。
- 综合以上方法进行合并。

Hack的避免

- 当避免的代价较大时，可以使用Hack而不避免，比如你需要增加很多HTML或多写很多CSS时会得不偿失。
- 丰富的实战经验可以帮助你了解那些常见问题并用多种不同的思路来避免它，所以经验和思维方法在这里显得很重要。
- 根据你自己的能力来解决Hack的问题，我们不建议你用一个自己都没有把握的方法来避免Hack，因为也许你这个方法本身存在你没有发现的问题。

如果CSS可以做到，就不要使用JS

让CSS做更多的事，减轻JS开发量。

- 用CSS控制交互或视觉的变化，JS只需要更改className。
- 利用CSS一次性更改多个节点样式，避免多次渲染，提高渲染效率。
- 如果你的产品允许不兼容低版本浏览器，那么动画实现可以交给CSS。

便于阅读修改

如果你做到了“CSS规范”的所有要求，自然你也就写出了一个便于阅读和修改的漂亮的CSS。

当然，代码格式和命名规则是相对重要一些的。

清晰的CSS模块

如果你做到了命名规则的要求，你的CSS模块也就清晰可见了。

用“注释”来说明每一个模块对于较大的CSS文件来说显得尤为重要。

文件压缩

合理的书写CSS能很大程度上减少文件大小，完成后，在不损坏文件内容的情况下，想尽一切办法压缩你的CSS，你可以借助压缩工具把注释和多余的空格、换行去掉。

其他格式优化

优化方法请参见代码格式。