【Sass中级】使用Sass和Compass制作雪碧图

作者：大漠 日期：2014-05-27 点击：9556

Preprocessor Sass SCSS Compass Sass中级

本文由大漠根据Aleksandar Goševski的《Spriting with Sass and Compass》所译，整个译文带有我们自己的理解与思想，如果译得不好或有不对之处还请同行朋友指点。如需转载此译文，需注明原作者相关信息http://thesassway.com/intermediate/spriting-with-sass-and-compass。

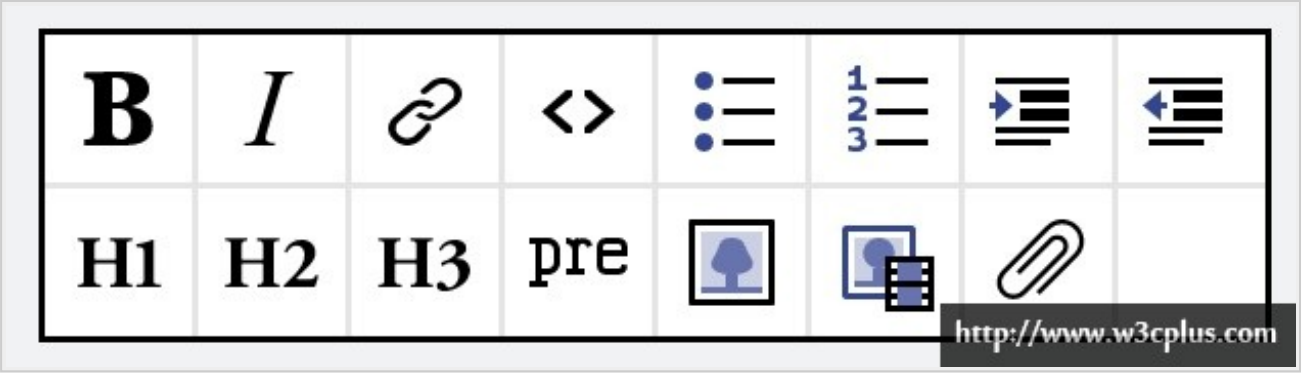
——作者：Aleksandar Goševski

——大漠

作为一名Web开发人员，在关注浏览器性能的时候，雪碧图("image spriting")这样的技术诞生了，旨在减少HTTP的请求数。事实证明，更少的请求数（当文件大小没有显著的区别时）对于一个页面的加载速度有明显的区别。

“Image Spriting”的工作原理是一堆的图像（称为“sprites”,精灵）合并成一张大的图像（国内称为雪碧图），以达到减少HTTP的请求数。然后通过 `background-position` 巧妙的显示雪碧图中需要呈现的图像。

下图是一个工具栏的雪碧图：



鉴于上面的图片，我们可以为媒体图标这样写样式：

```
$icon-width: 24px;
$icon-height: 24px;
$icons: image-url('toolbar.png');

.media-icon {
  background-image: $icons;
  background-position: -($icon-width * 5) -($icon-width * 1);
  width: $icon-width;
  height: $icon-height;
}
```

这样做是媒体图像是背景图中X轴方向的第五个之后，Y轴第一个图像之后：

雪碧图是出名的难维护。添加一个新的图像需要更新图像与相关的CSS。更糟糕的是如果你要删除一个图像时，会变得更为复杂。你会怎么做呢？重新做过一张雪碧图？

Compass来拯救你

幸运的是，Chris Eppstein的Compass项目包括了一套强大的工具，用于自动创建和维护雪碧图。Compass可以创建雪碧图，给出每个图的精确坐标，还可以让你控制图的布局下间距，并在SCSS中写入需要的图像。总之，Compass中制作雪碧图的工具，将节省你大量的时间与精力。

我并不想从头开始介绍Compass，因为这是项目浩大的工程，况且官网已经有很多教程。如果你并不熟悉Compass，我建议你先阅读这些教程先。

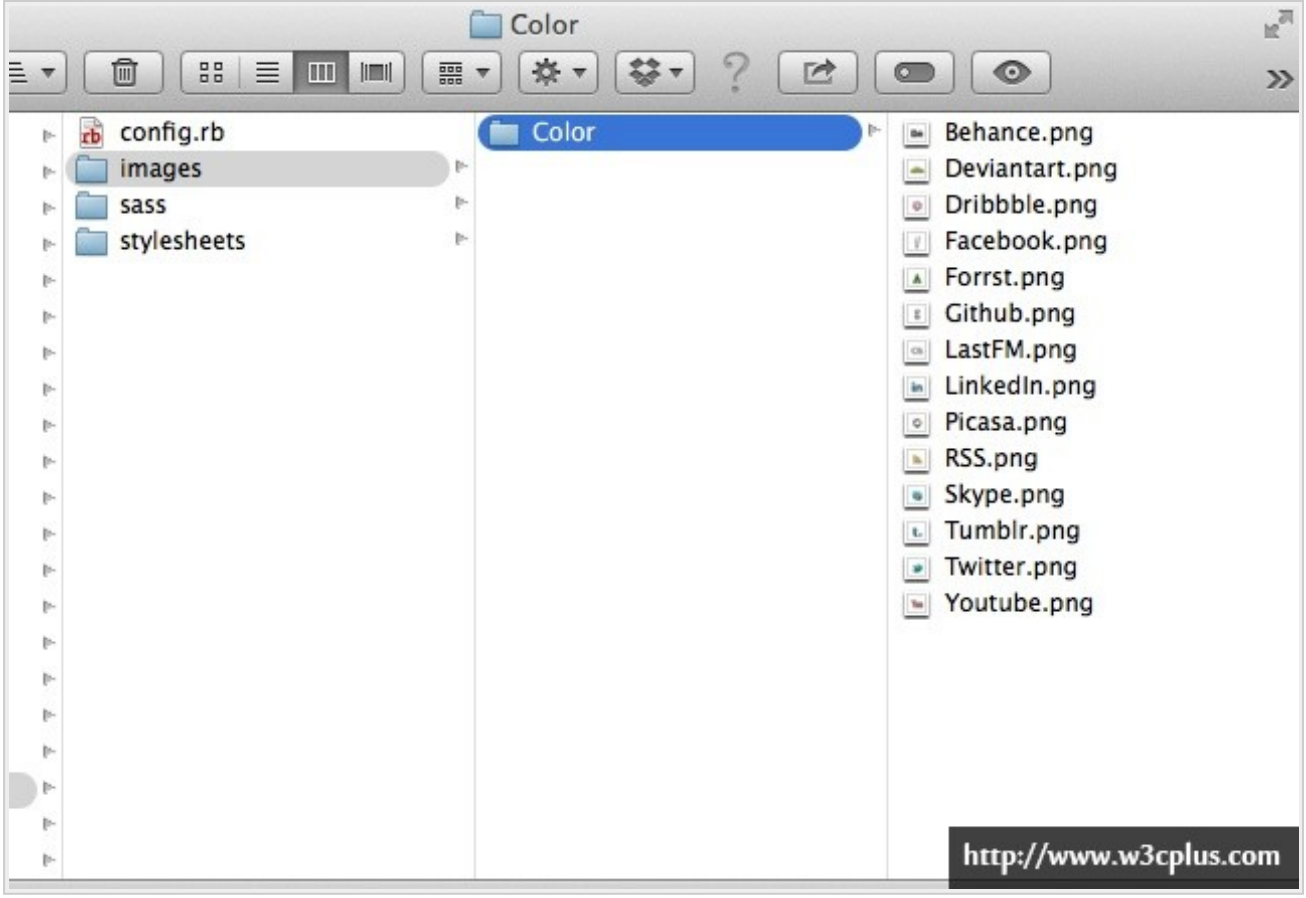
目录结构

根据Compass制作雪碧图的基本原理，你把图像放在一个文件夹中，而且这个文件夹放在`images/`的目录下，Compass会根据您提供的源图片生成一张雪碧图。对于我们工具栏的例子，我将图片源都放在了`images/toolbar`目录下，就像下面这样：

```
images/
|
`-- toolbar/
    |-- bold.png
    |-- italic.png
    |-- link.png
    |-- code.png
    |-- unordered-list.png
    |-- ordered-list.png
   ...
```

请记住，你应该只把需要的图片源放到这个文件夹内。Compass会利用这些图片源合并出你最图需要的雪碧图。

> 为了能更好的通过示例演示Compass和Sass实现雪碧图，将原文中的示例换成下图所示：（为了不去找图片源，我使用了我电脑中的一些图片以示说明）



最简单的方法

制作雪碧图最简单的方法就是使用Compass的 `@import` 命令：

```
@import "images/toolbar/*.png";
```

> 如果你的Sass更新到了最新版本(Sass 3.3.7 (Maptastic Maple))，那么运行上面的命令将无法实现，在命令终端会报错误信息。这个时候你只需要在命令终端运行： `gem install compass --pre` 。使用 `compass -v` 命令查看你的版本号是不是：Compass 1.0.0.alpha.19。如果无误，我们可以继续往下。
>
> 下面内容是译者实战中的经验：

为了能更好的实战Compass和Sass制作雪碧图，将原文中的结构换成了上图的效果，从图中可以看出，我们所有 `*.png` 放在一个名叫“**Color**”的文件目录之下，而且这个文件夹是放置在“ `images/` ”之下。如果按照原文教程所言，在 `.scss` 文件中直接通过 `@import` 命令即用：

```
@import "images/toolbar/*.png";
```

根据示例所示，我们只需要把 `toolbar` 换成我们的 `Color`：

```
@import "images/Color/*.png";
```

开启 `compass watch` 命令，终端会提示：

```
>>> Compass is watching for changes. Press Ctrl-C to Stop.
     info sass/screen.scss was modified
overwrite stylesheets/ie.css
overwrite stylesheets/print.css
    error sass/screen.scss (Line 8: No files were found in the load path matching
overwrite stylesheets/screen.css
```

文件路径错误，按照我们写CSS的经验，我将路径做相应的调整：

```
@import "../images/Color/*.png";
```

命令检测到：

```
info sass/screen.scss was modified
identical stylesheets/ie.css
identical stylesheets/print.css
   remove images/Color-s36a4fadee6.png
   create images/Color-s1760dc49ac.png
overwrite stylesheets/screen.css
```

虽然不报错，但看编译出来的 `.css` 文件，不难发现路径存在问题：

```
.Color-sprite {
    background-image: url('/images/../images/Color-s1760dc49ac.png');
```

```
    background-repeat: no-repeat;
}
```

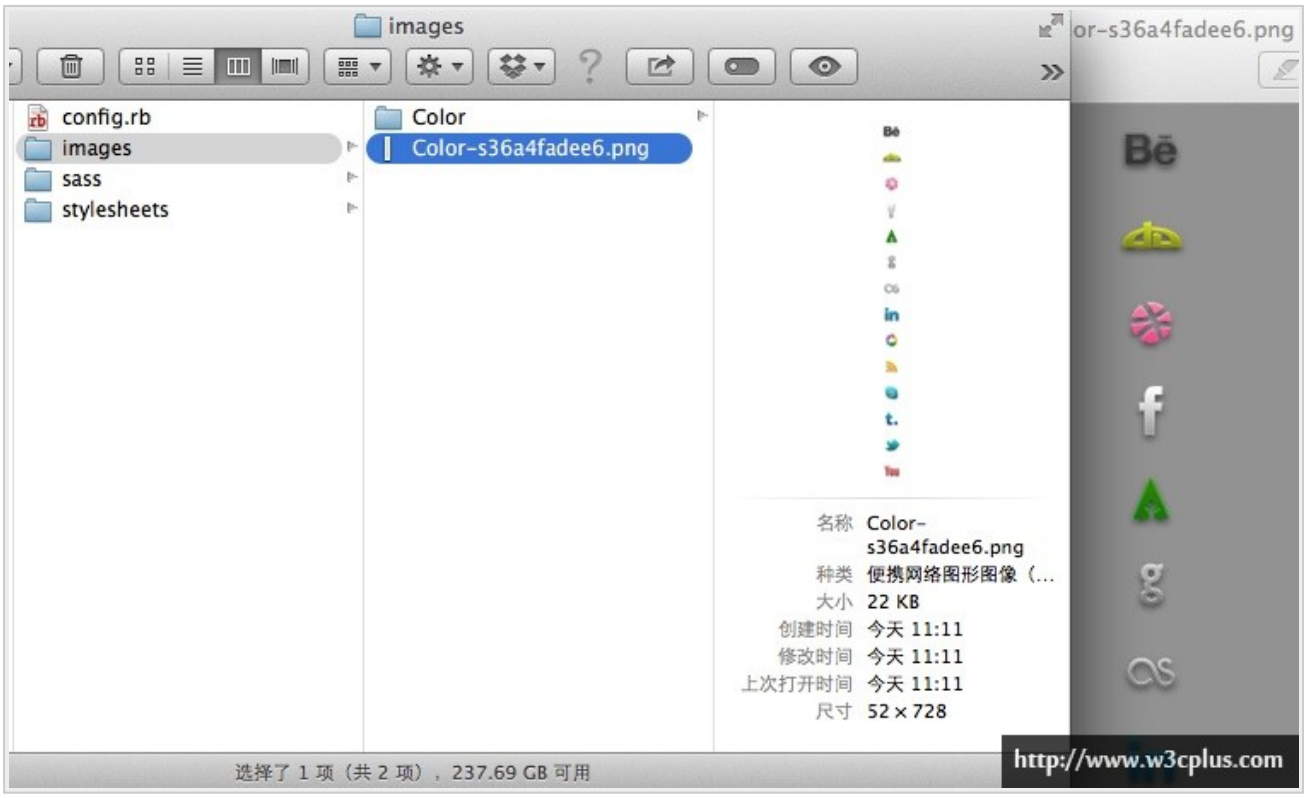
说实在的，这让我困惑。后来，我在想，是不是Compass已具备此功能，我们只需要将路径改成包含图片的文件夹开始，于是我尝试这样写：

```
@import "Color/*.png";
```

命令终端也不路径错误，而且编译出来的CSS也是我想要的：

```
.Color-sprite {
    background-image: url('/images/Color-s36a4fadee6.png');
    background-repeat: no-repeat;
}
```

此时在你的项目的"images/"可以看到一张名为“`Color-s36a4fadee6.png`”的图，如下所示：



大家可能会感到非常神奇，想知道为什么？那么我们接下来回到原文。

在Compass看到`@import`指令的参数为`*.png`时，它会假定将这个目录下的所有`.png`图片制作成一张雪碧图。让他生成一个mixin，使您在项目中更好的使用雪碧图。

其中mixin可以为雪碧图的所有图像生成对应的类。对于mixin的名称是基于引入图的文件夹名。例如我们的示例：

```
@include all-toolbar-sprites;
```

编译出来的CSS：

```
.toolbar-sprite, .toolbar-bold, .toolbar-italic, .toolbar-link {
  background-image: url('../images/toolbar-s1f1c6cbfd0.png');
  background-repeat: no-repeat;
}

.toolbar-bold {
  background-position: 0 0;
}

.toolbar-italic {
  background-position: 0 -24px;
}

.toolbar-link {
  background-position: 0 -48px;
}
```

请注意，Compass为我们自动创建了一张“toolbar-s1f1c6cbfd0.png”图片。这就是我们的雪碧图。这命名我们放图像的文件夹(在这个例子中叫toolbar)加上一串字母和数字。每当你更新图片源时，缓存的CSS就知道，并且会更新雪碧图。

> 我们再次回到我实战的用例中来（是不是感觉蛮乱的，有点神游）。按照原文的教程所言，我在实际用例中是这样做的：

在 `.scss` 文件通过 `@include` 调用Compass生成的mixin：

```
@include all-Color-sprites;
```

输出的CSS代码：

```
.Color-sprite, .Color-Behance, .Color-Deviantart, .Color-Dribbble, .Color-Facebook
  background-image: url('/images/Color-s36a4fadee6.png');
  background-repeat: no-repeat;
}

.Color-Behance {
  background-position: 0 0;
}

.Color-Deviantart {
  background-position: 0 -52px;
}

.Color-Dribbble {
  background-position: 0 -104px;
}

.Color-Facebook {
  background-position: 0 -156px;
}

.Color-Forrst {
  background-position: 0 -208px;
}

.Color-Github {
  background-position: 0 -260px;
}

.Color-LastFM {
  background-position: 0 -312px;
}

.Color-LinkedIn {
```

```
  background-position: 0 -364px;
}

.Color-Picasa {
  background-position: 0 -416px;
}

.Color-RSS {
  background-position: 0 -468px;
}

.Color-Skype {
  background-position: 0 -520px;
}

.Color-Tumblr {
  background-position: 0 -572px;
}

.Color-Twitter {
  background-position: 0 -624px;
}

.Color-Youtube {
  background-position: 0 -676px;
}
```

控制类名

如果你想更好的控制输出，不使用混合宏 `all-{文件夹名称}-sprites` 。在Compass你也可以使用单独的单独的混合宏。

```
@import "images/toolbar/*.png";

.bold-icon { @include toolbar-sprite(bold); }
.italic-icon { @include toolbar-sprite(italic); }
.link-icon { @include toolbar-sprite(link); }
```

这些混合宏同样是根据雪碧图的名称命名的。在我们的示例中“`toolbar-sprite`”。

> 在很多时候，我们希望调用的图片是根据需要调用雪碧图上的图像。Compass非常的强大，除了可以通过混合宏`all-{文件夹名称}-sprites`一次生成所有图像的CSS（类名是`Color-图像源文件名`）之外，还可以通过混合宏`{文件夹名称}-sprites({图像源文件名})`来实现按需加载，并且自定义类名。我们来看个简单的示例：

```
@import "Color/*.png";


.icon-twitter {
    @include Color-sprite(Twitter);
}
.icon-facebook{
    @include Color-sprite(Facebook);
}
.icon-youtube{
    @include Color-sprite(Youtube);
}
```

输出的CSS:

```
.Color-sprite, .icon-twitter, .icon-facebook, .icon-youtube {
  background-image: url('/images/Color-s36a4fadee6.png');
  background-repeat: no-repeat;
}

.icon-twitter {
  background-position: 0 -624px;
}

.icon-facebook {
  background-position: 0 -156px;
}

.icon-youtube {
```

```
  background-position: 0 -676px;
}
```

雪碧地图（Sprite maps）

如果你真的需要一个更低级的，Compass同样可以满足你，他提供了另一种工具——**雪碧地图(Sprite maps)**。让你在内部控制你的雪碧图。

使用雪碧地图，我们就不再使用 `@import` 指令了，是这样使用的：

```
$icons: sprite-map("toolbar/*.png");

.bold-icon { background: sprite($icons, bold); }
.italic-icon { background: sprite($icons, italic); }
.link-icon { background: sprite($icons, link); }
```

请注意，没有使用生成的雪碧图的混合宏，而是使用生成的雪碧地图的混合宏 `sprite` ，在对应的类名上插入图像。

> 同样，我们自己动手实战一下雪碧地图的功能。根据示例，我们也做一定调整：

```
$icons: sprite-map("Color/*.png");

.icon-twitter {
    background: sprite($icons,Twitter);
}
.icon-facebook{
    background: sprite($icons,Facebook);
}
.icon-youtube{
    background: sprite($icons,Youtube);
}
```

生成的CSS如下：

```
.icon-twitter {
  background: url('/images/Color-s29211bcaaa.png') 0 -624px;
}

.icon-facebook {
  background: url('/images/Color-s29211bcaaa.png') 0 -156px;
}

.icon-youtube {
  background: url('/images/Color-s29211bcaaa.png') 0 -676px;
}
```

控制间距

很多时候，我们制作雪碧图时，每个图像之间需要有一定的空白距离。在实际应用中，这一点也是非常重要的。

比如在每个icon四周设置一个间距：

```
// 使用@import
$toolbar-spacing: 5px;
@import "toolbar/*.png";

// 使用Sprite Maps
$icons: sprite-map("toolbar/*.png", $spacing: 5px);
```

> 在我们实际用例中，我们每个icon的大小是52px*52px。为了更好的适应偶数计算，我希望每个icon之间有一个8px的间距。根据上面介绍，我们可以这样做。

先来看 `@import` 的方式：

```
$Color-spacing: 8px;
@import "Color/*.png";

.icon-twitter {
```

```
    @include Color-sprite(Twitter);
}
.icon-facebook{
    @include Color-sprite(Facebook);
}
.icon-youtube{
    @include Color-sprite(Youtube);
}
```

这个时候Compass会重新生成一张雪碧图，我们将有无间距的两张雪碧图来对比一下：

特别注意，使用`@import`指令，我们定义变量时需要以`{文件夹名称}-spacing:间距值;`格式来定义，如果你的变量名和放置图片源的文件夹名称不匹配，将无法生成带有间距的

雪碧图。

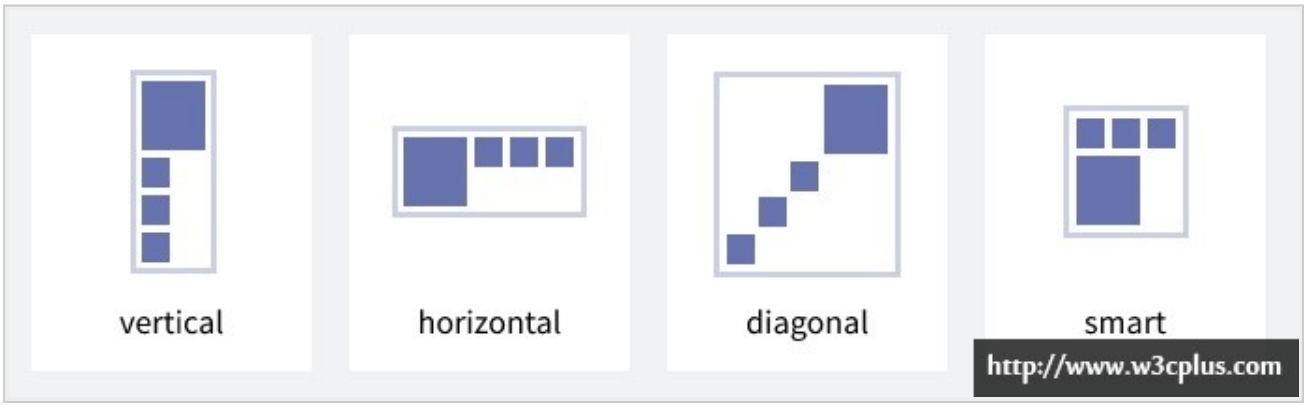
接下来看第二种，就是雪碧地图的方式：

```
$icons: sprite-map("Color/*.png",$spacing:8px);
```

这种方式与第一种方式生成的雪碧图是一样的。

控制布局

Compass支持几种不同的布局方式，就是雪碧图中icon的排列方式：



包括四种排列方式：vertical、horizontal、diagonal和smart。其中vertical为其默认的排列方式。

在不同的生成雪碧图方式中，其使用方式也略有不同。也就是`@import`和雪碧地图，设置方式不同：

```
// 使用@import
$toolbar-spacing: 5px;
$toolbar-layout: 'smart';
@import "toolbar/*.png";

// 使用雪碧地图Sprite Maps
$icons: sprite-map("toolbar/*.png", $spacing: 5px, $layout: diagonal);
```

注意：在当前版本，你不能同时使用间距和智能(`smart`)布局。

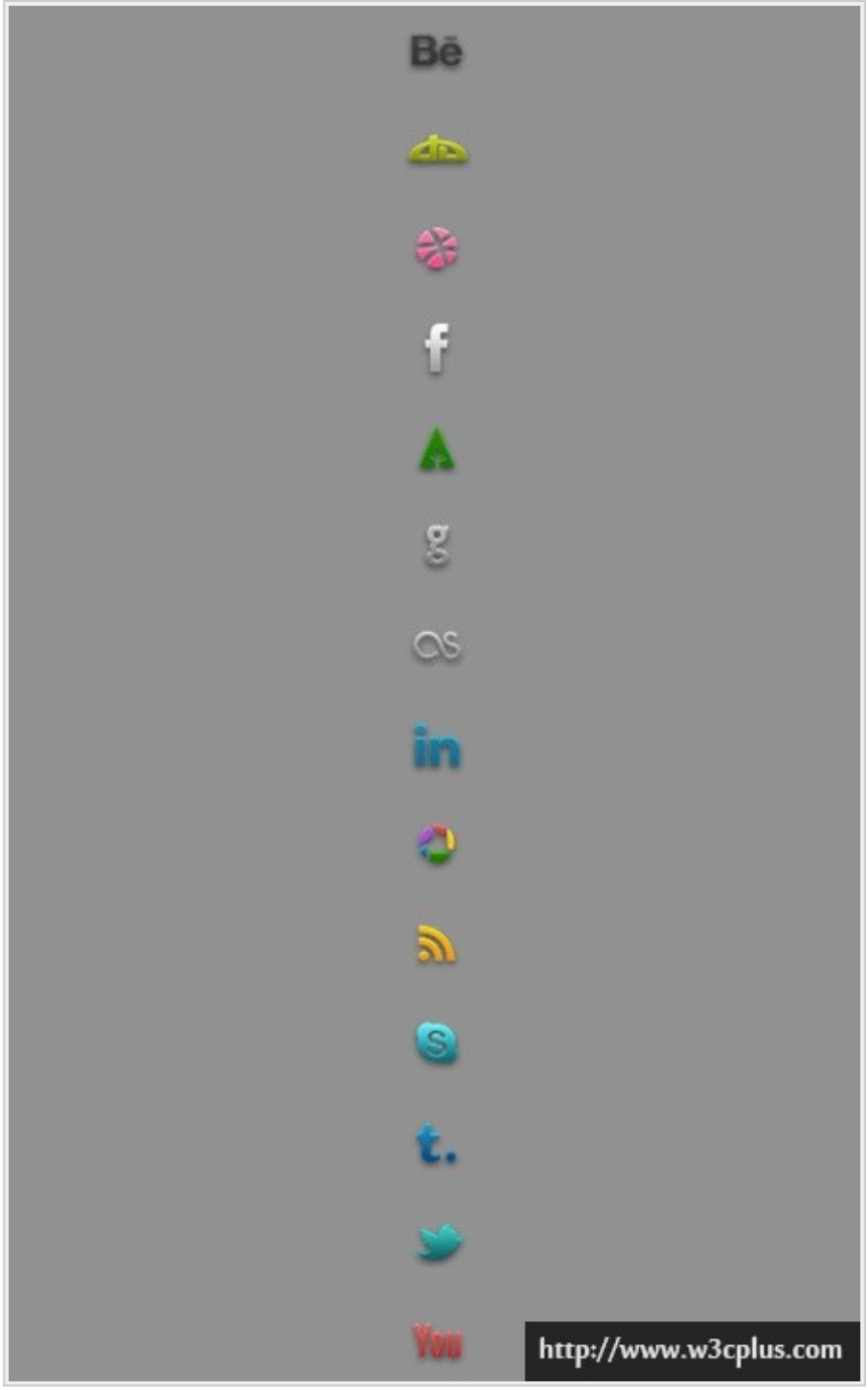
> 我们接下来亲自实战一下这四种排列图标的效果。

垂直排列

使用Compass生成雪碧图，其图像中的icon排列方式是按垂直方式排列，这种排列方式也是其默认的排列方式，不需要显式的声明。当然，你显示的声明也是可以的。如：

```
$Color-spacing: 8px;
$Color-layout: "vertical";
@import "Color/*.png";
```

生成的雪碧图如下：



水平排列

按照上面的方式，将 `$Color-layout` 变量的值设置为“**horizontal**”。

```
$Color-spacing: 8px;
$Color-layout: "horizontal";
@import "Color/*.png";
```

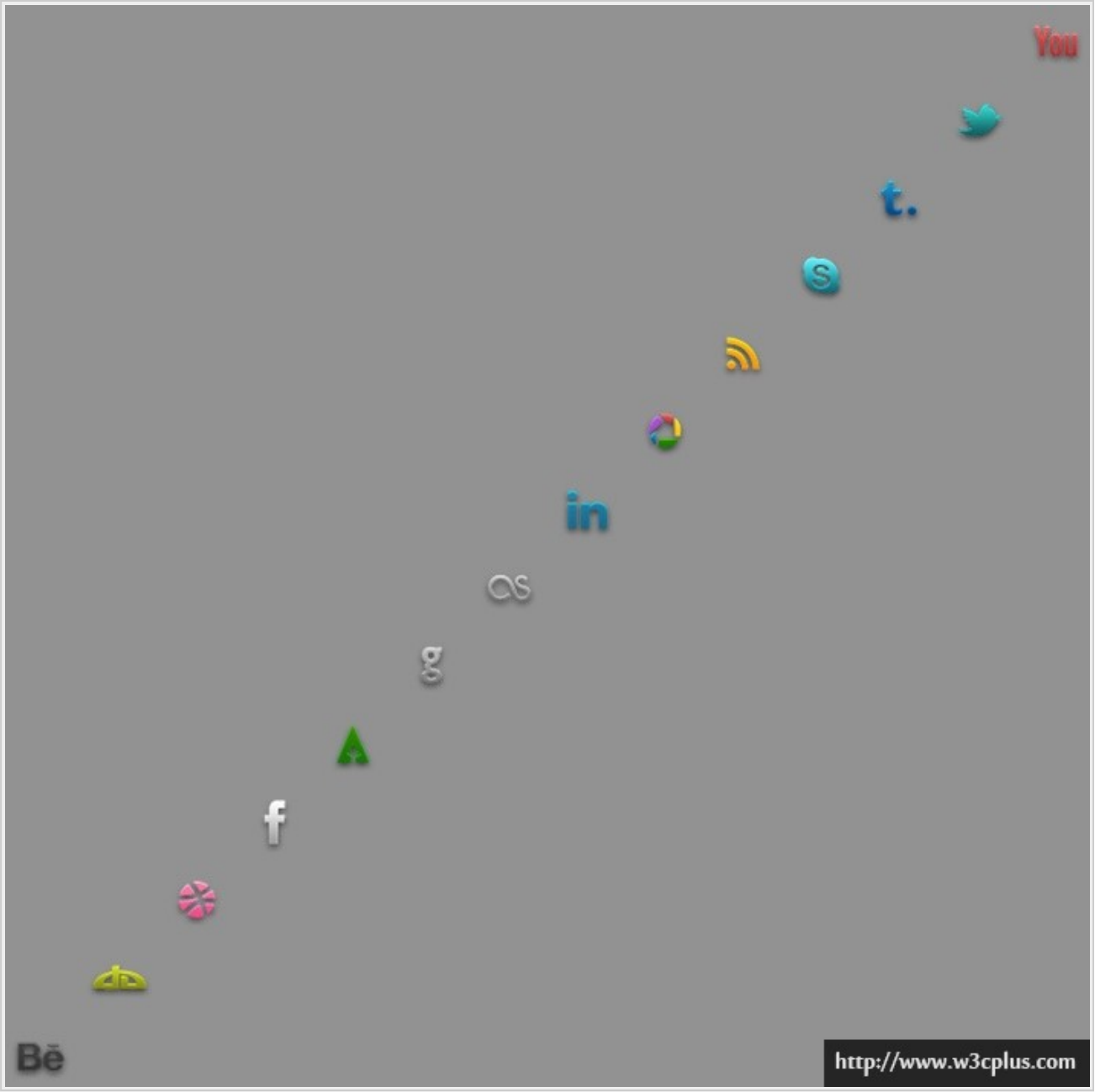
生成的雪碧图如下所示：

斜角排列（成45度角）

同样的将 `$Color-layout` 变量的值设置为“**diagonal**”。

```
$Color-spacing: 8px;
$Color-layout: "diagonal";
@import "Color/*.png";
```

生成的雪碧图如下所示：



智能排列

将 `$Color-layout` 值设置为"**smart**":

```
$Color-spacing: 8px;
$Color-layout: "smart";
@import "Color/*.png";
```

可以生成的雪碧图与默认垂直的效果是一样的。我在想是不是哪操作失误。前面有说过间距和智能排列不能同时出现，虽然我的Compass是 `pre` 版本，我还是尝试禁用了间距的设置。可还是依旧。效果跟前面示意完全不同。我在想是不是因为icon大小都一样，我不仿修改一下加过一张不同类型的图片到源文件夹中。

这个时候，重新看看生成出来的雪碧图效果：

效果出来了。

特别提醒：使用Compass设置图标排列方式时，定义变量的方式和定义间距变量方式类似，需要和对应的源图的文件夹结合起来。比如我们此处所有图片源都放在`Color`的文件夹中，那么定义的变量名是：`Color-layout`。也就是`${文件夹名称}-layout:[vertical][horizontal][diagonal][smart]`。

上面演示的是通过`@import`指令生成雪碧图的排列方式，那么在雪碧地图中使用方法类似，你只需要修改参数`$layout`的值即可。

```
$icons: sprite-map("Color/*.png",$spacing:8px,$layout:smart);
```

其它函数和混合宏功能

Compass还为雪碧地图提供了一些其他的函数(`function`)和混合宏（`mixin`）功能:

- `sprite-url($icons)`：返回雪碧图的URL
- `sprite-position($icons,bold)`：返回“bold”图标在雪碧图中的X轴和Y轴的坐标值
- `@include sprite-dimensions($icons,link)`：根据雪碧图的原始尺寸设置like图标的宽度和高度。

使用方法如下:

```
$icons: sprite-map("toolbar/*.png", $spacing: 5px, $layout: diagonal);
.bold-icon {
  background-image: sprite-url($icons);
  background-position: sprite-position($icons, bold);
  @include sprite-dimensions($icons, bold);
}
```

> 根据上面所述，亲自体会了一回:

```
$icons: sprite-map("Color/*.png",$spacing:8px,$layout:smart);

.icon-twitter {
    background-image: sprite-url($icons);
    background-position: sprite-position($icons,Twitter);
    //@include sprite-dimensions($icons,Twitter);
}
.icon-facebook{
    background-image: sprite-url($icons);
    background-position: sprite-position($icons,Facebook);
    //@include sprite-dimensions($icons,Facebook);
}
.icon-youtube{
    background-image: sprite-url($icons);
    background-position: sprite-position($icons,Youtube);
    //@include sprite-dimensions($icons,Youtube);
}
.icon-ctrip{
    background-image: sprite-url($icons);
    background-position: sprite-position($icons,Ctrip);
    //@include sprite-dimensions($icons,Ctrip);
}
```

大家都看到了，我把`@include sprite-dimensions()`功能给注释掉了，因为在我的环境中没有编译成功。目前我也还没有查出切确原因。希望有知道原因的同学希望能分享一下。最后编译出来的CSS:

```
.icon-twitter {
  background-image: url('/images/Color-s6ea6182c52.png');
  background-position: -156px 0;
}

.icon-facebook {
  background-image: url('/images/Color-s6ea6182c52.png');
  background-position: 0 0;
}

.icon-youtube {
  background-image: url('/images/Color-s6ea6182c52.png');
  background-position: 0 -52px;
```

```
}

.icon-ctrip {
  background-image: url('/images/Color-s6ea6182c52.png');
  background-position: 0 -208px;
}
```

更深层次的挖掘

使用Compass制作雪碧图，其实还有很多，信不信由你。以上只是一个简单的教程。如果你对这方面感兴趣，可以阅读官网提供的教程，或者从这里查阅其他文档。

扩展阅读

- Stop Making Sprites (Compass, Sass, and PNG Sprite Generation)
- Automating Semantic Sprites with Compass
- How to make icons for the Web [part 2]
- 使用Compass生成雪碧图

译者手语：整个翻译依照原文线路进行，并在翻译过程略加了个人对技术的理解。如果翻译有不对之处，还烦请同行朋友指点。谢谢！

如果文章中有不对之处，烦请各位大神拍正。如果你觉得这篇文章对你有所帮助，打个赏，让我有更大的动力去创作。(^_^)。看完了？还不过瘾？点击向作者提问！