## CSS雪碧

CSS雪碧 即[CSS Sprite](http://baike.baidu.com/view/2173476.htm)，也有人叫它CSS精灵，是一种CSS图像合并技术，该方法是将小图标和背景图像合并到一张图片上，然后利用css的背景定位来显示需要显示的图片部分。

## 原理

CSS雪碧的基本原理是把你的网站上用到的一些图片整合到一张单独的图片中，从而减少你的网站的HTTP请求数量。该图片使用CSS background和background-position属性渲染，这也就意味着你的标签变得更加复杂了，图片是在CSS中定义，而非<img>标签。

一个简单的例子：

[](http://baike.baidu.com/picture/3037553/3037553/0/fc5e5f34d01c6920251f14a1.html?fr=lemma&ct=single)一张图片作出一个按钮的三个状态

一个链接用CSS做成按钮的样式，我们可以使用同一张图片，完成按钮的三个状态，a:link，a:hover，a:active。

<a class="button" href="#">链接</a>

加入右侧的图片为：200px×65px的三个按钮图拼合而成的图片button.png，从上到下一次为按钮的普通、鼠标滑过、鼠标点击的状态。则可以使用CSS进行定义。

a {

display:block; width:200px; height:65px; line-height:65px; /\*定义状态\*/

text-indent:-2015px; /\*隐藏文字\*/

background-image:url(button.png); /\*定义背景图片\*/

background-position:0 0; /\*定义链接的普通状态，此时图像显示的是顶上的部分\*/}

a:hover {background-position:0 -66px; /\*定义链接的滑过状态，此时显示的为中间部分，向下取负值\*/}

a:active {background-position:0 -132px; /\*定义链接的普通状态，此时显示的是底部的部分，向下取负值\*/}

初学的朋友可以试试效果。

而更多的CSS雪碧，图片更复杂，背景定位更精确。可能会用到大量的数值，如：background:url(nav.png) -180px 24px no-repeat; 来达到更精确的定位。

## 优点

### 减少加载网页图片时对服务器的请求次数

可以合并多数背景图片和小图标，方便在任何位置使用，这样不同位置的请求只需要调用一个图片，从而减少对服务器的请求次数，降低服务器压力，同时提高了页面的加载速度，节约服务器的流量。

### 提高页面的加载速度

sprite 技术的其中一个好处是图片的加载时间(在有许多 sprite 时，单张图片的加载时间)。由所需图片拼成的一张 GIF 图片的尺寸会明显小于所有图片拼合前的大小。单张的 GIF 只有相关的一个色表，而单独分割的每一张 GIF 都有自己的一个色表，这就增加了总体的大小。因此，单独的一张 JPEG 或者 PNG sprite 在大小上非常可能比把一张图分成多张得来的图片总尺寸小。

### 减少鼠标滑过的一些bug

IE6不会主动预加载鼠标滑过即a:hover中的背景图片，所以，如果使用多张图片，鼠标滑过会出现闪白的现象。使用CSS雪碧，由于一张图片即可，所以不会出现这种现象。

## 不足

### CSS雪碧的最大问题是内存使用

除非这个雪碧图片是被非常小心的组织，你就会最终使用大量的无用的空白。一个例子是来自于WHIT TV的网站。注意这是一个1299×15,000像素的PNG图片。它也被压缩的很好——实际下载大小只有大概26K — 但是浏览器并不会渲染压缩后的图片数据。当这个图片被下载并被解压缩之后，它将占用差不多75MB的内存 (1299 \* 15000 \* 4)。如果这个图片并没有使用alpha透明，它将会被优化至1299 \* 15000 \* 3，但是要在损失渲染速度的情况下。即使那样，我们也会讨论55MB。这张图片的大部分其实就是**空白**，那里什么都没有，没没有任何有用的内容。只是加载 WHIT主页 就会导致你的浏览器的内存占用上升到至少75+MB，仅仅是因为那一张图片。（PS:遗憾的是，该网站最近已经改版，文中提到的图片已经不存在了）

### 影响浏览器的缩放功能

如果一个使用CSS雪碧的页面使用一些浏览器提供的整页缩放功能缩放了，浏览器就需要做一些额外的工作来纠正这些图片边缘的行为——基本上来说，是为了避免雪碧中相邻的图片被“露进来”。这对于小图片没有什么问题，但是对于大图片会是一个性能下降。

### 拼图维护比较麻烦

拼合这么多图片，需要耐心。同时还要时刻思考如何在使用这个图片是不会产生相互的影响。将又瘦又高的图片和又宽又矮的图片放到一起时，不容易操作。如果要修改雪碧中的一个图片，你就要修改整张图片，这无疑会增大工作量。

### 使CSS的编写变得困难

如果CSS雪碧足够复杂，则大大增加了CSS的代码量和难度，让维护和修改变得困难起来。

### CSS 雪碧调用的图片不能被打印

CSS 雪碧调用的图片不能被打印，除非在@media中特别添加 print声明。

### 错误得使用 Sprites 影响可访问性

一些刚入门的开发人员会为了节省 HTTP 请求数(这是使用 CSS Sprite 一直强调的好处)而把所有的图片都当背景图片来处理 – 甚至是那些传达重要信息的图片。结果会导致一个缺乏可访问性的网站，也会降低 HTML 中 title 和 alt 的潜在益处。

因此，CSS sprite 本身没错，而且也不会引发可访问性问题(事实上，正确得使用会提高可访问性)。但是不分对错得过度使用 sprite 会阻碍具有可访问性和生产率方面的网页建设进程。

## 实用技巧

### 不要等到你完成切片之后才开始sprite

如果你边切图边写CSS，然后等你完成了整个网站之后再来拼接这些图片到一个Sprite中，你就不得不完全重写你的CSS，你也必须要花费很多的时间来用PS拼接大量的图片——这是件令人倍感纠结的事情。但是如果边切图边整合，就会比较容易些。

### 把图片放到它要显示的地方的相对的地方

这个小技巧貌似比较难理解。我直到创建一个比较大的sprite的时候才理解到这一点。比如，如果我们希望一个图片出现在一个元素的左侧：将那个图片放到sprite图片的右边(本文开始的那个sprite图片)。这样的话，当你通过CSS移动背景图片的位置的时候，基本上不可能有其它的小图片意外的出现在它的附近。使用Sprite的时候常常遇到的一个问题是图片会出现在它不该出现的位置。

### 定位时避免使用bottom或right等

当使用CSS sprite的时候，只用background-position: bottom -300px或background-position: right -200px;非常容易。这刚开始的时候是可行的，但是问题是，当你在宽度上或高度上扩展相关sprite图片的时候，原先设置的位置可能是错的，因为那个图片已经不再Sprite图片的底部或右部了。使用确切的位置来避免这个问题。

### 给每个图片足够的空间

就像你在本文顶部的实例图片看到的那样，那些小图片都被预留了足够的空间。为什么不把他们塞到一块来让sprite图片更小呢？ 因为使用这些图片的元素通常都会有大量的内容而且可能会需要扩展的间距，以至于其它图片不会意外出现。

### 不用担心Sprite图片的像素大小

如果你的网站经过良好的设计，那么你将会有一大堆的图片来整合进到sprite里面，这样你就需要你个非常大的sprite来恰当的放置这些图片。这是很不错的。sprite里的空白不会占用太多的文件大小。国外某购物网站上使用的Sprite图片有1,000px×2,000 px那么大，但是图片的大小仅仅**16.7kb!**

链接：

<http://baike.baidu.com/link?url=4ttY6BxID2d4yFza8CCAHEvCN2GybYsW72zUDHn21uI05VRBGfadl9tzUG-nuK38Gb0z3cQM4N4BJnjvzLtM_K>