**8. Image Sprite.**

**What are CSS Sprites?**

**Spoiler alert: they aren't fairies that write your stylesheets for you. I wish.** In short: CSS Sprites are a means of combining multiple images into a single image file for use on a website, to help with performance.

Sprite may seem like a bit of a misnomer considering that you're create a large image as opposed to working with many small ones, but the [history of sprites](http://en.wikipedia.org/wiki/Sprite_%28computer_graphics%29), dating back to 1975, should help clear things up.

To summarize: the term "sprites" comes from a technique in computer graphics, most often used in video games. The idea was that the computer could fetch a graphic into memory, and then only display parts of that image at a time, which was faster than having to continually fetch new images. The sprite was the big combined graphic.

CSS Sprites is pretty much the exact same theory: get the image once, and shift it around and only display parts of it. This reduces the overhead of having to fetch multiple images.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-1)**Why use CSS Sprites?**

It may seem counterintuitive to cram smaller images into a larger image. Wouldn't larger images take longer to load?

Let's look at some numbers on an actual example:

| **Image** | **File Size** | **Dimensions** |
| --- | --- | --- |
| canada.png | 1.95KB | 256 x 128 |
| usa.png | 3.74KB | 256 x 135 |
| mexico.png | 8.69KB | 256 x 147 |

That adds up to a total of 14.38KB to load the three images. **Putting the three images into a single file weighs in at 16.1KB. The sprite ends up being 1.72KB larger than the three separate images.** This isn't a big difference, but **there needs to be a good reason to accept this larger file... and there is!**

While the total image size (sometimes) goes up with sprites, **several images are loaded with a single HTTP request**. Browsers limit the number of concurrent requests a site can make and HTTP requests require [a bit of handshaking](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#HTTP_session).

Thus, sprites are important for [the same reasons](https://developers.google.com/speed/docs/insights/MinifyResources) that minifying and concatinating CSS and JavaScript are important.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-2)**How do you use CSS Sprites?**

Here's an example sprite, with three different countries flags combined into a single image:



You set the same background-image on several CSS classes and set the background position and dimensions of the individual classes to display a single portion of the sprite. Here's some code that demonstrates the concept:

.flags-canada, .flags-mexico, .flags-usa {

background-image: url('../images/flags.png');

background-repeat: no-repeat;

}

.flags-canada {

height: 128px;

background-position: -5px -5px;

}

.flags-usa {

height: 135px;

background-position: -5px -143px;

}

.flags-mexico {

height: 147px;

background-position: -5px -288px;

}

If you're thinking that there has to be a way to automate this so that you aren't manually creating these sprites and then adjusting your stylesheet to match, you're right, and you're in luck!

[**#**](https://css-tricks.com/css-sprites/#article-header-id-3)**Generate Sprites with Grunt / Gulp / Node**

If you're using Grunt, Gulp, or Node in general, (now called [sprity](https://www.npmjs.com/package/sprity)) is a wonderful node package that creates sprites from a glob of images. Sprity has a lot of great features including formatting output as PNG, JPG (or [Data URIs](http://css-tricks.com/data-uris/) of those), and stylesheet generation in CSS, LESS, Sass, and Stylus.

To compile sprites via command line, install css-sprite globally with:

$ npm install sprity -g

Then, to generate sprites and the corresponding stylesheet, run:

$ sprity ./output-directory/ ./input-directory/\*.png

For more information on using css-sprite with Grunt or Gulp (or many other enviornments), head over to [the project's repository on GitHub](https://github.com/aslansky/css-sprite).

[**#**](https://css-tricks.com/css-sprites/#article-header-id-4)**Generate Sprites with Compass**

[Generating sprites with Compass](http://compass-style.org/help/tutorials/spriting/) takes some additional setup and maintenance, but if you're already using Compass, it fits in well with your existing workflow.

Start by creating a directory within your `images` directory (yes, it does need to be inside your `images` directory to work) with a name that corresponds to the sprites you'd like to create. Ensure that the images you're converting to sprites are PNGs and place them in your new directory. I'm creating flag sprites, so I've named my directory flags and placed three PNGs in the directory.

In a new SCSS file that I've called `flags.scss` (the name here is not important), the following three lines will, in order, import Compass' sprite making tools, glob import the PNGs to be converted to sprites (notice that the path here does not include images/), and then generate the CSS for the sprites. Be mindful that **the @include statement's middle word needs to match the directory in the line before it**.

@import "compass/utilities/sprites";

@import "flags/\*.png";

@include all-flags-sprites;

Here's an example of Compass output, using our flags example:

This is a fairly simple process for generating sprites, but it has a few drawbacks/oddities:

* The generated CSS does not include widths or heights for the sprites.
* There is no shared class between the sprites; the background-image is applied to each class.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-5)**Generating Sprites with ImageMagick**

[ImageMagick](http://www.imagemagick.org/) can be used to create a spritesheet from the command line with the following commands:

convert \*.png -append sprites.png # append vertically

convert \*.png +append sprites.png # append horizontally

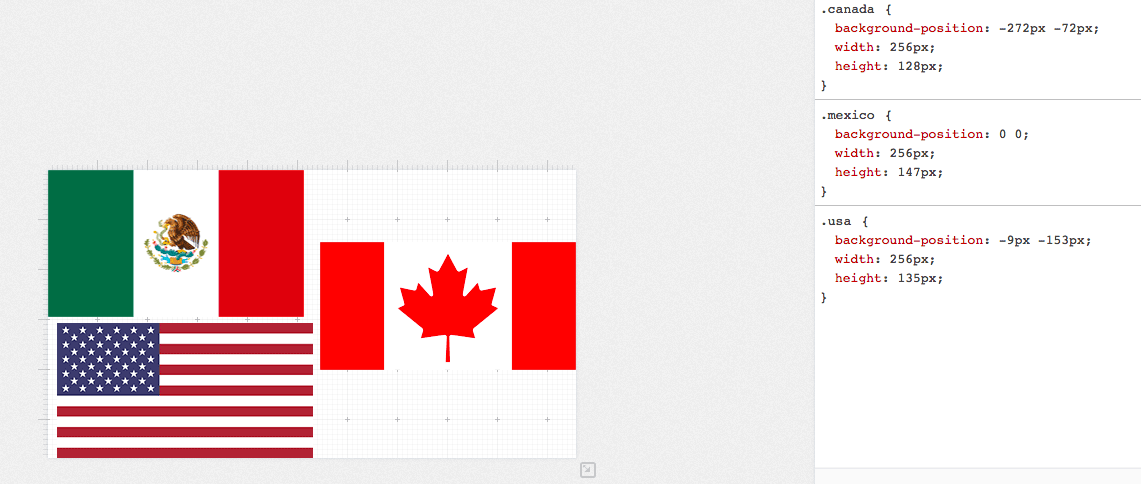
This will take all the PNG files selected by the glob and concatenate them into a single file, but will not create the corresponding stylesheet. If you use ImageMagick to create your sprites, you may want to read the section below on using Sprite Cow.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-6)**Using Sprite Cow with your Sprites**



[Sprite Cow](http://www.spritecow.com/) is a hosted tool for generating a stylesheet corresponding to your sprites. It doesn't make the sprite for you, it just helps you get numbers you need to use the sprite (the width, height, and background-position of individual parts of the sprite). It boasts 2x image compatibility and a simple interface for quickly designating which areas of the sprite make up each image to create CSS for. You just click the part you need and it gives you the CSS you need.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-7)**Generate Sprites with Spritepad**



[Spritepad](http://wearekiss.com/spritepad) is another hosted solution for creating sprites. With Spritepad, you upload individual images, position them however you'd like, and the CSS is updated in real time. When you're done, download the image and copy the CSS over to your project.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-8)**Generate Sprites with SpriteMe**

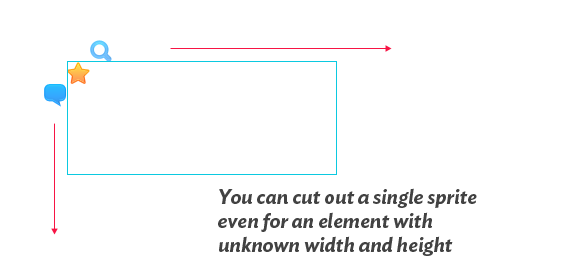
[SpriteMe](http://spriteme.org/) is a bookmarklet that generates a sprite based on what it finds on the current page. So essentially you'd develop without using sprites at all, then use this to sprite things together at the end. [Here's a workflow](http://css-tricks.com/css-sprites-workflow/) explaining how that would work.

[**#**](https://css-tricks.com/css-sprites/#article-header-id-9)**Should my sprites be horizontal or vertical?**

One option is neither. Compact them into a grid, making the smallest size, dimensionally that you can. The dimensional size of an image plays a role in how much memory the image will take up when being used, so the less the better. If you end up laying out your own sprite, Sprite Cow is a good tool to help with the CSS generation part.

If, for simplicity, you're going to pick one or the other, one way to do that is to look at the largest width and the largest height of your image files. If the largest width is greater than the largest height, the sprite sheet should be arranged horizontally. If the largest height is greater than the largest width, vertically. If you're using a generation tool, they will generally make this choice for you.

In some situations, it may actually make sense to [lay out a sprite diagonally](http://www.aaronbarker.net/2010/07/diagonal-sprites/). This can make it possible to use a sprite in an area of unknown width and height, which is pretty cool.



Although another possible way to get around that is [using a pseudo element](http://css-tricks.com/pseudo-spriting/).