

2.6: Graphics

Images are the greatest contributors to the visual appeal of your site, and typically account for the majority of bandwidth used in loading your pages. By using a combination of image types, and newer techniques found in HTML5 like canvas, and reproducing images using CSS, we can balance quality against size to reduce our bandwidth needs and allow our site to be more flexible.

Formats

Images are files, just like any other document in your computer, but they can be coded and formatted differently to reproduce the image you want to see. We find these referred to as raster and vector graphics. These formats represent two very different methods of creating an image.

Raster

The image files most of us are already familiar with using are typically raster format. Examples of these are JPEG, GIF and BMP. When we interact with pictures we took on digital cameras for example, we are dealing with JPEG or JPG files. Raster files recreate an image by recording the color value of pixels, which represent the smallest single point on a screen that can be assigned a color by the display. The higher the number of pixels (or density, measured as pixels per inch) translates to how sharp the image is, and how large it can be rendered without losing quality.

The number of colors available in the image file is based on the length of the value available to each point. If we only allowed a single binary character for each pixel point, we would be able to keep our file size as small as possible. This however would mean we could only represent our image in black and white (binary only allows us two options, 0 or 1, so we can only represent two colors.). When we allow longer values to represent a single point, we can assign values a larger range of colors. Once we scale these up, however, we trade away our smaller image sizes in order to have more colorful pictures. Large images can slow down the user experience, and if loading takes too long, users will leave.

Traditionally, we have faced this trade off by using different image formats in different areas of our site. While reserving JPG for our larger images or photos, we can use GIF for smaller icons and indicators. GIFs limit us to 256 colors, but since most icons use few colors, we are able to capitalize on the benefits of this format here. It is important to note that raster images will quickly lose quality when rendered at sizes larger than the original image's width or height.

Vector

Vector images store information about colors and locations as definitions of angles, lines, and curves in mathematical format. The benefit of a vector formatted image is that it can be scaled both up and down in size without distortion or degradation in quality. This is due to the fact that the image is “drawn” by the browser each time it is loaded, and the processor performs the steps necessary to recreate the image. Since the image can be scaled, the same image file can be drawn very large, or very small, without changing the file size. We will get some hands-on experience in how vector images are drawn when we look at the new [Canvas](#) features in [HTML](#).

Table 2.6.1: Image Formats

| Format | Compression | Platforms | Colors | Notes |
|--|-------------|----------------|--|---|
| JPEG (Joint Photographic Experts Group) | Lossy | Unix, Win, Mac | 24-bit per pixel; 16.7 million colors. | JPEG is a compression algorithm; the format is actually JFIF (JPEG File Interchange Format) |

| | | | | |
|------------------------------------|--------------------------|----------------|--|---|
| GIF (Graphic Interchange Format) | Lossless | Unix, Win, Mac | 8-bit; 256 colors (216 web palette). Allows transparency. | LZW compression algorithm developed by CompuServe; patent now held by Unisys, which charges for use of the code in graphics programs. Once Unisys began enforcing its patent (in 1995), programs began moving to PNG. |
| BMP (Bitmap graphics) | Uncompressed | Win | 24-bit; 16.7 million colors. | Like all uncompressed formats, these files are very large. |
| PICT | Lossless | Mac | | Very little compression; large files |
| TIFF (Tag Interchange File Format) | Lossless or uncompressed | Unix, Mac, Win | | TIFF-LZW uses the proprietary LZW compressions (see GIF). |
| PNG (Portable Network Graphics) | Lossless | Unix, Mac, Win | 48-bit; “true color” plus transparency | Will likely replace GIF. Supported in IE, NN 4 and above. A WC3 specification . |

<http://mason.gmu.edu/~dtaciuch>

You may notice the compression column. This is the act of removing or modifying the data that represents a file in a manner that makes its overall file size smaller. By doing this, we can transmit files faster, and they will take up less space in memory. When we discuss compression in terms of graphics we need to consider whether it will result in a lossy or lossless result. A lossless result means the compression techniques used do not remove data from the original copy, so we can restore the image to its exact original size and appearance. A lossy compressions structure can result in greater compression, but achieves the extra advantage by removing information from the file.

As an example, imagine a picture of you and your friends on the beach with a clear blue sky behind you. The data in the image file will measure the “blueness” of the sky in varying colors of blue, at a level greater than the eye can distinguish. By averaging the blueness and making more of the sky pixels the same color, we have eliminated information. Certain levels of lossy compression will still be indistinguishable from the original, but at any level, the lossy-compressed version of the file will not be restorable to the original because the extra values were eliminated.

Which is better? As usual, it depends on your intent. If the image can be lossy compressed and is still acceptable to you and your users, and having the smallest possible file sizes (good, of course, for mobile devices) is a priority, then go for it. Quality optimized scenarios will likely call for a lossless compression, like in sites that use large images as their background.

Slicing

For some time, there has been a practice of breaking larger images up into many smaller ones (a process called slicing), in an effort to allow pages to load more quickly. While this gave a visual experience of faster speed (each small image blinking into place as it was loaded) the load time was about the same, if not longer, as the overall file size had not changed, but we instead asked for it over multiple requests instead of waiting for the entire image in one request.

The need for this approach has been largely eliminated by modern versions of CSS (and other techniques we will discuss). This allows us to reproduce many things we used images for (i.e. buttons, hover effects, etc.) without using images at all, and allows us to have the control over layout and formatting that slicing an image used to fulfill. Now a common goal for site developers is to be

as “imageless” as possible, using images only where CSS cannot stand in. This reduces load times and gives greater flexibility in site design. As an example of what can be done using CSS3, take a look at this simulated iPhone: <http://tjrus.com/iphone#71d465!>

Some additional techniques to reduce image weight on a site are right-sizing images, compression (which we just discussed above), caching, and sprites, among others. Right-sizing is editing and creating a copy of an image to the exact size needed where it is shown. For example, small images for items on a product page could simply be the original image rendered at a smaller size. If the user does not look at those products, loading the larger images first only degrades their experience, since every product’s large image file needs to be downloaded. Right-sizing and compression both require image editing software with at least some advanced editing features, or use of an online service that covers the basics (with less control on your part) like <http://www.imageoptimizer.net>.

The process of caching can also help. When your site is completed and your images (and other files) will persist, the use of caching can reduce load times for repeat visitors. Caching means the files the user downloads are marked in the server with an expiration. The next time the user visits the site, their device will check expiration times on the content. If the device’s local copy is not expired, it simply uses the one it has, without having to download it again.

The last option we will consider here (there are more, advanced methods) is creating a sprite. Almost the reverse of compression and right sizing, a sprite is one large image that contains all of the icons used throughout the site. Since these smaller images are often repeated, we will download one main image a single time. By using CSS rules to reveal only small piece of the image (i.e. the portion containing one icon) at a time, all of our icons can live in one image file but appear as individual elements on the page.

Favicon

A favicon is a special type of image. It is the small icon that accompanies the page title in the browser’s title and tab. This icon is automatically used on each page found in the folder the favicon is stored in. For example, to apply a single icon to your entire site you would place it in the root folder. Any folders below that level can use a different favicon.

These icons are usually 32 by 32 pixel images that represents the site or site’s parent company. They are converted to a special format and saved with the extension .ico to identify themselves as site icons. While they are small and provide little to the overall visual experience of a website, sites lacking a favicon tend to appear less legitimate as the icon space will be replaced by the browser’s default icon.

Creating favicons can be done in paint or photo editing software that allows you to comply with the size and color density limits of favicons. Additionally, sites like favicon-generator.org or www.favicon.cc among others can help convert existing images into favicons with some basic editing options before saving your new icons.

2.6: Graphics is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- 2.6: Graphics by [Michael Mendez](#) has no license indicated.