A LIST APART

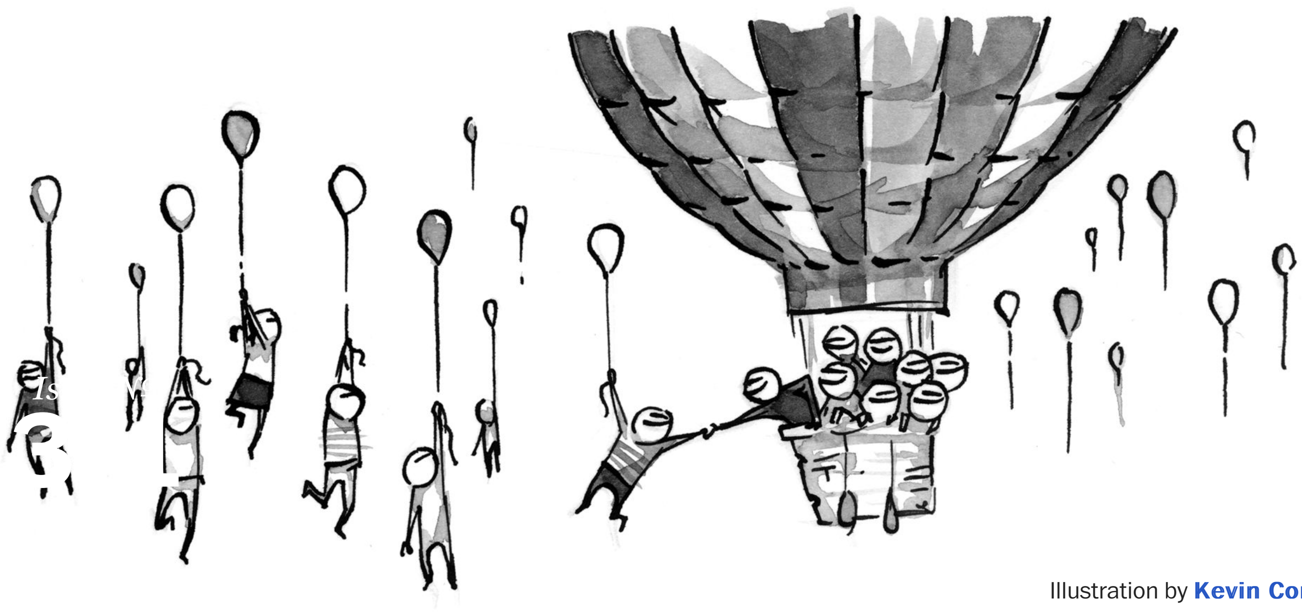


Illustration by **Kevin Cornell**

The Era of Symbol Fonts

by **Brian Suda** · March 12, 2013

Published in Graphic Design, Typography & Web Fonts

Improving performance is a constant process. First we ditched tables, spacer gifs, and inline markup such as the `<font>` element in favor of CSS, reducing page sizes, and separating style from layout. Then we became aware of all our DNS requests, caching, and the total number of files and started using CSS sprites, moving many small images out of the HTML and into a single background image.

Now it's time we embrace the third epoch in performance optimization: symbol fonts.

Embedding a symbol font lets us move some of those tiny icons into a single font file rather than a sprite. This has the same caching and file size benefits as a CSS sprite, as well as some additional benefits we're only now realizing with high-resolution displays. In this article, I'll walk you through some of the advantages and issues you'll encounter when using a symbol font.

A smooth experience

As the number of fonts designed for use as icons, glyphs, and decorations increase, several high-traffic websites have replaced all the images on their site with a single symbol font. One of the best-known examples is GitHub, which has improved both its speed and its user experience by dropping all its tiny icon files in favor of Octicons *(https://github.com/blog/1106-say-hello-to-octicons)*, a single, custom symbol font file.

Symbol fonts trade blocky raster images for smooth vector images in your HTML. This is important because rasterized images are designed to work at a specific resolution, so at higher pixel densities, like those found in today's high-resolution displays, the images must be scaled up to appear the same size. This results in something that looks more "blocky." Vectors never have this problem, scaling endlessly up or down for any resolution.

Embedding vector graphics into HTML has had limitations in the past. PDF and SVG are possible, but create interoperability problems. Fonts are vectors and all web browsers, even IE6, have the ability to render embedded fonts in an HTML page. This convergence of technologies—font embedding, high-resolution displays, and browser support—has created a new opportunity to further optimize performance and user experience.

Accessibility

When it comes to accessibility, symbol fonts are in the conflicted situation of working poorly as images, yet quite well as text.

Many of the early symbol font services mapped the graphic to a letter in the ASCII range. Typing a "w" would produce a picture of a globe, typing an "m," an envelope. This made it easy to see the character-to-symbol conversion simply by typing. Problems would occur when the font embedding failed, ending up with strange letters in random places that didn't make sense. For instance, if your markup was meant to display an arrow, then the text "Next," the HTML might look like:

```
<div><span class="icon">L</span> Next</div>
```

But if the CSS, JavaScript, or font loading were to fail, you would see the character "L" rather than the symbol for a left arrow. You could put all the fallbacks in place, inject the code with JavaScript or CSS `:before` or `:after`, but at the end of the day, search engine spiders might still index that text as "L Next."

There is a better way.

We've matured, learned more about these issues, and solved the problem by using the private-use area in Unicode *(http://en.wikipedia.org/wiki/Private_Use_(Unicode))*. The glyphs in this area are not connected to the semantics of any letters of the alphabet. If the font isn't available or doesn't load, then an empty box will appear instead of a letter.

But wait—we can improve even more!

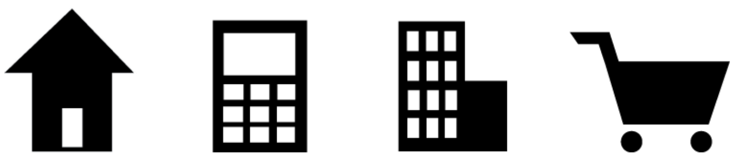
Fonts also have special characters called ligatures—subtle, often hardly noticeable, tweaks to the letterforms that are used to aid reading. Take, for instance, two consecutive "f" characters. A good font will convert that "ff" into a single ligature where the fs connect smoothly. There are several standard ligatures, including ff, fl, and fi. But there is no reason you can't also define your own. In a font file, it is a simple substitution; all the ligature is looking for is the right sequence of letters. When they're typed, they are replaced with another glyph. This means you can have a string like "A List Apart" and convert it into a single symbol icon of the logo.

Browsers that don't support font embedding—namely, search engine spiders—will index the raw un-ligatured text, but devices capable of rendering the font and ligature will show you the nice vector symbol instead. It's the best of both worlds, baked in at a much higher level than HTML and CSS.

Once you understand how ligatures work, a world of new possibilities opens up! All those mystery-meat icons used in navigation could be symbols in a font, but beyond that they could be ligatures. So your HTML markup might look like:

```
<ul>
   <li>Home</li>
   <li>Contact</li>
   <li>About</li>
   <li>Cart</li>
</ul>
```

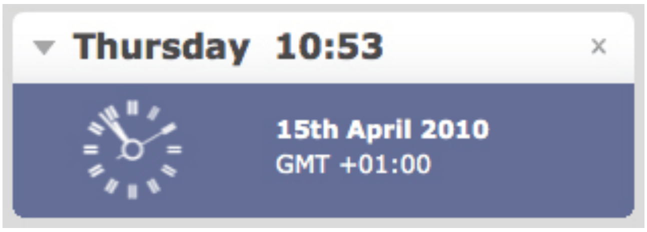
But the resulting rendered ligatures might look like:



Symbolset *(http://symbolset.com)* has been experimenting with ligatures and symbol fonts for a while, and now sells packs of symbol fonts with commonly used icons. It also has great ligature support, and a demo where you can simply type keywords and watch them snap to symbols immediately. Don't be surprised if this makes your mind race with ideas.

Using ligatures as glyph substitutes has practical applications. For example, the BBC News website formerly featured a ticking clock *(http://www.bbc.co.uk/blogs/webdeveloper/2010/04/good-news-the-clock-has.shtml)* in the corner of the homepage. The clock could have used Flash, but that would have been ill-advised given its lack of support and the existence of alternatives. Another option would have been to use CSS to rotate images of the hands with a little JavaScript timing. Instead, the BBC chose HTML5, using Canvas plus JavaScript to continually redraw the clock face.

But both these non-Flash solutions use rasterized images that wouldn't look as nice on high-resolution displays. There are JavaScript libraries, like Raphaeljs, which can create SVG or VML objects that would be nice and vectorized, but since they are loaded and created in JavaScript, they would be invisible to search engine spiders or other non-JavaScript browsers.



Using a symbol font instead would give you the best of both worlds: accessibility and vectors. Typing 12:00 would call up a ligature, which would convert to a clock with both hands at 12. Typing 12:01 would be a different ligature for the corresponding clock face. While there would be a lot of ligatures to create, the resulting clock would be a beautifully designed vector graphic that would work across platforms. In fact, a few clock typefaces are already doing this; http://timepiece.inostudio.de *(http://timepiece.inostudio.de)* has several examples. Go ahead—view source, and you'll see its simplicity. *[Ed. note: We restored the link to inostudio.de after hearing from the site owner that the issue had been addressed, and after running our own scan.]*

Think about all a web page's little details that could be symbol fonts. Open your mind to the possibilities. Everything from bullets and arrows to feed and social media icons could be bundled into a single, tiny font file that can be cached and rendered at various sizes without needing multiple images or colors.

Creating your own symbol font

Web developers and designers cut their teeth on applications like Photoshop, Fireworks and others. For us, moving to CSS sprites or background images wasn't a big leap because the tools didn't change. With symbol fonts, a new program enters into the mix: the font editor. This is a new beast we need to learn, but it's not that difficult.

Even better, plenty of tools are being released to aid in symbol font creation. For example, Icomoon *(http://icomoon.io/)* lets you upload your SVG files, map them to unicode characters, and export a font for use online.

What if you want to get more specific and tweak the design beyond its default displays? There are several tools out there to create your own fonts. A few are paid, professional apps, but the most common free tool is FontForge *(http://fontforge.org/)*. It's a bit unwieldy, but there are many tutorials out there to help you use it.

Issues with fonts

As any good designer will tell you, an icon or logo will have subtly different designs at different sizes, such as changing its line thicknesses or even dropping some of its detail at small sizes. With a symbol font, however, you can't have alternative designs for different font sizes. Your design at 20 point must be the same as at 120 point or 6 point.

If you want different designs at different font sizes, then you need to duplicate the icon in the font file and select the appropriate glyph depending on the usage. This too is problematic, because you have no idea how the end user will actually view your design. Increasing the font size in their browser natively won't switch your symbol design. For most of us, this isn't a problem, as many of the utility icons we use today for feeds and social networking scale relatively well.

The other major issue with using a symbol font is that you only get one color. You can work some magic with background colors and font gradients to emulate a two-color logo, but if your icons are multicolored, then the current symbol font setup won't work for you.

To remedy this, Apple is proposing multicolor fonts, which would let you create and embed pure vector graphics into any web page. Its first foray into this field is the Apple Color Emoji *(http://en.wikipedia.org/wiki/Apple_Color_Emoji)* font, which you might already have if your machine runs OSX Lion. The downside of introducing a new font format is the lack of browser support and font creation tools needed to come up to speed, which means it might be a while before this becomes a reality.

Welcome to the next epoch

Symbol fonts are the next step forward in website design and optimization, and you need to understand them to work on the web today. With practice, you can create your own custom symbol fonts with logos, icons, and branding to embed into your websites. Not only will these little extras impress your audience, but they will simultaneously optimize your website's performance. It's time we all start using symbol fonts—and help improve the tools, standards, and techniques to make them.

About the Author

Brian Suda

Brian Suda is an informatician residing in Reykjavík, Iceland. He's written a book on the topic of charts and graphs entitled Designing with Data. His own little patch of internet is at suda.co.uk where many past projects and crazy ideas can be found.

MORE FROM THIS AUTHOR

SVG with a little help from Raphaël *(/article/svg-with-a-little-help-from-raphael)*
Enhance Usability by Highlighting Search Terms *(/article/searchhighlight)*

ISSN 1534-0295 · Copyright © 1998–2018 A List Apart & Our Authors