Quick guide to webfonts via @font-face

By Paul Irish

Published: August 2nd, 2010
Comments: 19

Introduction

The @font-face feature from CSS3 allows us to use custom typefaces on the web in an accessible, manipulable, and scalable way. But, you might say, "Why would we use @font-face if we have Cufon, sIFR, and using text in images?"

View the Font Previewer for a taste of how flexible webfonts are

A few benefits of leveraging @font-face for custom fonts:

- Full searchability by Find (ctrl-F)
- Accessibility to assistive technologies like screen readers
- Text is translatable, through in-browser translation or translation services
- CSS has full ability to tweak the typographical display: `line-height`, `letter-spacing`, `text-shadow`, `text-align`, and selectors like `::first-letter` and `::first-line`

@font-face at its essence

At its most basic, we declare a new custom remote font to be used like so:

```
@font-face {
  font-family: 'Tagesschrift';
  src: url('tagesschrift.ttf');
}
```

Then put it to use:

```
h1, h2, h3 { font-family: 'Tagesschrift', 'Georgia', serif; }
```

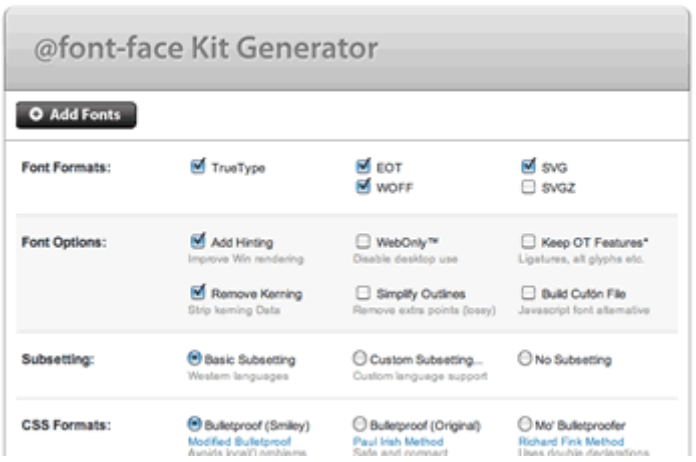
In this @font-face declaration we're using the `font-family` property to explicitly name the font. It can be anything, regardless of what the font is actually called; `font-family: 'SuperDuperComicSans';` would work out just fine, though perhaps not for your reputation. In the `src` property we point to where the browser can find the font asset. Depending on the browser, some valid font types are eot, ttf, otf, svg, or a data URI embedding the entire font data inline.

	OTF & TTF	SVG	WOFF	EOT
IE	IE9		IE9	IE5+
FIREFOX	FF3.5	FF3.5	FF3.6	
CHROME	Chrome 4	Chrome 0.3	Chrome 5	
SAFARI	3.1	3.1		
OPERA	Opera 10.00	Opera 9		
IOS		iOS 1		
ANDROID	2.2			

Of course, nothing is ever as simple as it should be. The initial limitation of the above code was that it did not serve an EOT to IE 6-8. The bulletproof @font-face syntax proposed a way of solving this; a robust version follows.

```
@font-face {
  font-family: 'Tagesschrift';
  src: url('tagesschrift.eot'); /* IE 5-8 */
  src: local('☺'),              /* sneakily trick IE */
       url('tagesschrift.woff') format('woff'),    /* FF 3.6,
Chrome 5, IE9 */
       url('tagesschrift.ttf') format('truetype'), /* Opera,
Safari */
       url('tagesschrift.svg#font') format('svg'); /* iOS */
}
```

Inducing a headache yet? If you'd rather just get this off the ground quickly, head to the Font Squirrel generator, a tool that simplifies the whole process for you, taking your font and preparing its variants and CSS for you. It's indispensable for putting webfonts into practice today.



The Font Squirrel generator

Mobile support?

Mobile Safari supports SVG webfonts as of iOS 3.1 and Android supports otf/ttf as of version 2.2. But should your mobile users get this enhanced typographic experience? I'd recommend no. The predominant reason is due to how WebKit handles text that is awaiting a custom font via @font-face: the text is invisible. So on a low-bandwidth mobile connection, your users will see no text at all until the ~50k of font data has loaded. The Webkit team is pursuing a solution of turning on a fallback font after a few seconds have expired, but until that has landed, I wouldn't consider it fair to subject your users to such roadblocks between them and your content.

Webfont services

A number of services wrap the @font-face feature in an easy API, oftentimes letting you add a single CSS or script line to your HTML and some configuration and you're all done. Many like WebInk, Typekit, and Fontslive will allow you to use the fonts (sometimes up to a bandwidth cap) for a monthly fee. Using these services is very convenient for the casual developer, handing off some of the complications of serving a cross-browser solution

The Google Font API lets you use a small, curated set of freely licensed fonts by just linking to a stylesheet and letting Google handle the cross-browser and performance concerns. It's the fastest way to get off the ground and running with webfonts.

Finding professional typefaces for @font-face

A common surprise to designers is that just because you purchase a font license (to use in your graphic design, for example), that doesn't mean you can use it in @font-face. Licenses for @font-face (or web embedding) are typically sold separately. Read the agreement carefully, and feel free to contact the font foundry if you have questions.

Fontspring is a font boutique, selling hundreds of quality professional fonts, all of them cleared for use with @font-face. FontFont, and other foundries, have begun selling @font-face licenses directly, though currently only targeting WOFF and EOT, which leave out a sizable (but shrinking) portion of the browser market. Many foundries are adding webfont licenses to their catalog, but if you can't find one for your chosen typeface, get in touch with them to ask about it.

Dealing with FOUT

The Flash of Unstyled Text is a phenomenon in Firefox and Opera that few web designers are fond of. When you apply a custom typeface via @font-face, there is a brief moment, when loading the page, where the font hasn't been downloaded and applied yet, and the next font in the `font-family` stack is used. This causes a flash of a different (typically less good looking) font, before it gets upgraded.

The Flash of Unstyled Text happening to the HTML5 Slide deck.

Accompanying the Google Font API is the WebFont Loader, a javascript library aiming to provide a number of event hooks giving you a lot of control over the loading. Let's take a look at how you can get other browsers to mimic WebKit's behavior of hiding the fallback text while the @font-face font loads.

```
<script
src="//ajax.googleapis.com/ajax/libs/webfont/1/webfont.js">
</script>
<script>
WebFont.load({
  custom: {
    families: ['Tagesschrift'],
    urls: ['http://paulirish.com/tagesschrift.css']
  }
});
</script>
```

```
/* we want Tagesschrift to apply to all h2's */
.wf-loading h2 {
  visibility: hidden;
}
.wf-active h2, .wf-inactive h2 {
  visibility: visible;
  font-family: 'Tagesschrift', 'Georgia', serif;
}
```

If JavaScript is disabled, the text will remain visible the whole time, and if the font errors somehow, it'll fall back to a basic serif safely. Consider this a stop-gap measure for now; most webfont experts desire hiding the fallback text for 2-5 seconds, then revealing it. Low-bandwidth and mobile devices benefit greatly by this timeout. Understandably, Mozilla is looking to rectify this soon.

A more lightweight but less effective solution is the `font-size-adjust` property, currently only supported in Firefox. It gives you an opportunity to normalize x-height across a font-stack, reducing the amount of visible change in the FOUT. In fact, the Font Squirrel generator just added a feature where it tells you the x-height ratio of the fonts you upload, so you can accurately set the `font-size-adjust` value.

Summary

Webfonts deliver quite a bit of freedom to designers and with upcoming features like discretionary ligatures and stylistic alternates, they will have a lot more flexibility. For now, you can feel confident implementing this part of CSS3 as it covers 98% of deployed browsers. Enjoy!