

# Introduction to variable fonts on the web



By Mustafa Kurtuldu

Design Advocate at Google • Material Design, UX & Design Sprints

In this article, we will look at what variable fonts are, how we can use them in our work, and the potential possibilities they entail. But to understand what they offer, first we must explore how typography and font loading currently work on the web.

## Introduction

The terms font and typeface are used interchangeably in the developer industry. However there is a difference: A typeface includes an entire family of designs, such as the Roboto typeface. Meanwhile a font is one of the digital files of that family, like "Roboto Bold" or "Roboto Italic." In other words, a typeface is what you see, and the font is what you *use*.

Roboto

**SUNGLASSES**

*Self-driving robot ice cream truck*

**Fudgesicles only 25¢**

**ICE CREAM**

Marshmallows & almonds

#9876543210

***Music around the block***

Summer heat rising up from the boardwalk

Thin  
*Thin Italic*  
Light  
*Light Italic*  
Regular  
*Regular Italic*  
Medium  
*Medium Italic*  
Bold  
*Bold Italic*  
Black  
*Black Italic*

Above: Roboto was designed and developed by Christian Robertson. On the left, a specimen of the Roboto typeface. On the right different fonts within the Roboto family.

## Challenges for the Designer and Developer

When a graphic designer prepares their work, they typically export the final artwork in a way that either embeds all of the fonts used, or they provide an export archive that contains all of the assets used separately, a bit like a basic website. The cost to the designer depends on the suitability of the typeface for the rendering context (is it legible at small sizes, for example) and the license to use the fonts.

On the web, we have to consider both aspects, plus the associated bandwidth costs. For every member of a typeface family used in our designs, we have had to require another font file to be downloaded by our users before they can see that text. Only including the Regular and Bold styles, plus their italic counterparts, can amount to 500k or more of data. This has been a sticking point for web designers and developers, as a richer typographic experience

comes at a cost. This is even before we have dealt how the fonts are rendered, and the fallback or delayed-loading patterns we are going to use (such as ["FOIT"](#) and ["FOUT"](#) [\[7\]](#)).

## Anatomy of a variable font

A variable font is a collection of master styles, with one central 'default' master (usually the Regular font style) and multiple registered "axes" which tie the central master to the other masters. For example, the **Weight** axis might connect a Light style master to the default style and through to the Bold style master. The individual styles that can be located along this axis are called instances.

For example, the variable font [Amstelvar](#) [\[8\]](#) has three masters for its **Weight** axis: The Regular master is at the center, and two masters, thinner and bolder, are at the opposite ends of the axis. Between these there are potentially 200 instances that can be chosen by the designer or developer:



Typeface Amstelvar, designed by David Berlow, type designer and typographer at Font Bureau.

The [OpenType specification](#) [\[9\]](#) specifies other axes, such as **Width**, **Optical Size**, **Italic** and **Slant**. All of these axes share the same default master, and we can combine them to explore an exponential number of typographic styles, like powers of 2, 3 and 4 do to numbers.

Amstelvar also has three masters in a Width axis: Again the Regular is at the center of the axis, and two masters, narrow and wider, are at the opposite ends of this axis. These not only provide all the widths of the Regular style, but also all the widths and weights combined.

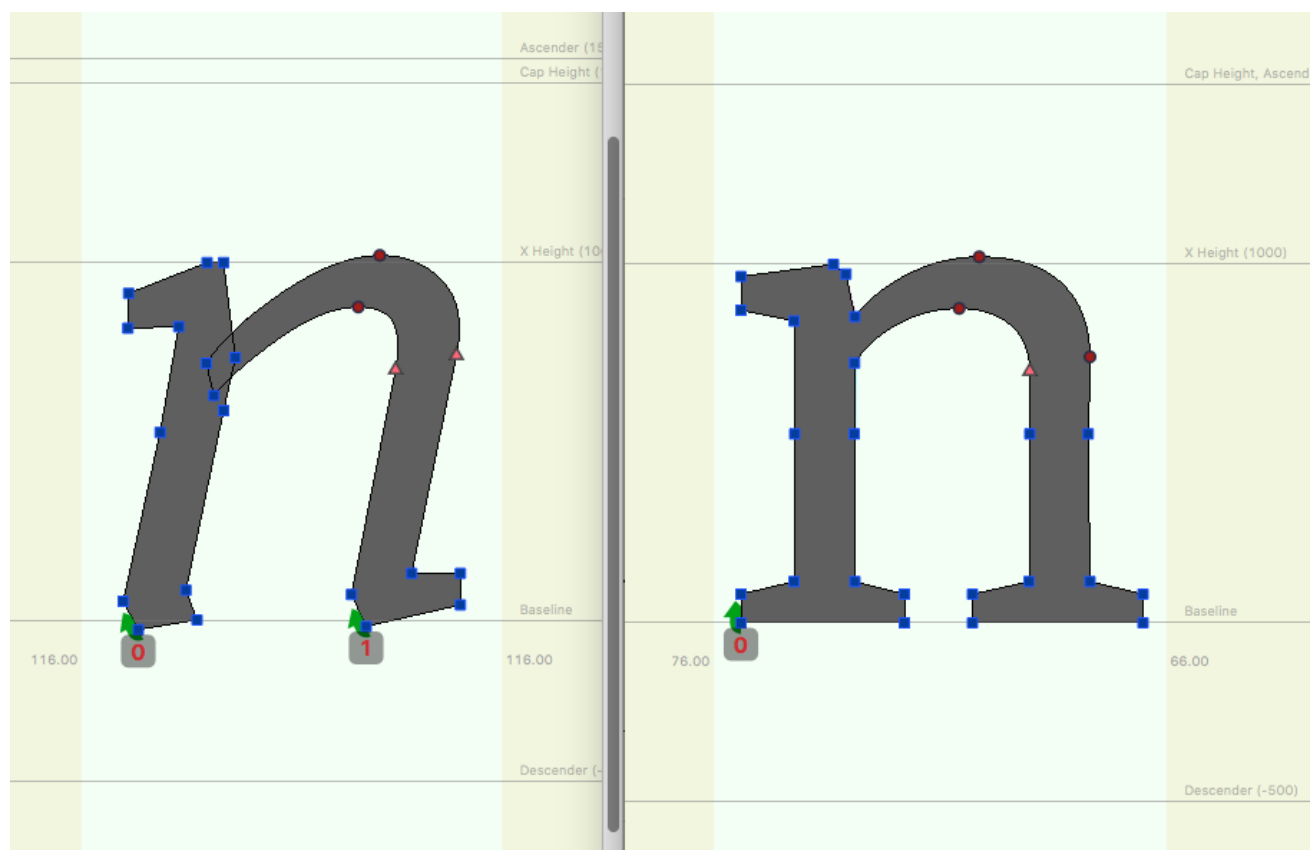
Among Amstelvar's registered axes (width, weight and optical size) there are thousands of styles. This may seem like massive overkill, but consider that Amstelvar only supports the Latin writing system. Considering the needs of all the world's scripts, and many of today's typography applications, the quality of the reading experience can be remarkably enhanced

by this diversity of type styles within a font. And, if it is without performance penalty, the user can use a few or as many styles as they wish – it's up to their design.

## Italics are slightly different

The way that Italics are handled in variable fonts is interesting, as there are two difference approaches. Typefaces like Helvetica or Roboto have interpolation compatible contours, so their Roman and Italic styles can be interpolated between and the **Slant** axis can be used to get from Roman to Italic.

Other typefaces (such as Garamond, Baskerville, or Bodoni) have Roman and Italic glyph contours that are not interpolation compatible. For example, the contours that typically define a Roman lowercase “n” do not match the contours used to define an Italic lowercase “n”. Instead of interpolating one contour to the other, the **Italic** axis toggles from Roman to Italic contours.



Amstelvar's “n” contours in Italic (12 point, regular weight, normal width), and in Roman. Image supplied by David Berlow, type designer and typographer at Font Bureau.

After the switch to Italic, the axes available to the user should be the same as those for the Roman, just as the character set should be the same.

A glyph substitution capability can also be seen for individual glyphs, and used anywhere in the design space of a variable font. For example, a dollar sign design with two vertical bars works best at larger point sizes, but at smaller point sizes a design with only one bar is better. When we have fewer pixels for rendering the glyph, a two bar design can become illegible. To combat this, much like the Italic axis, a glyph substitution of one glyph for another can occur along the **Optical Size** axis at a point decided by the type designer.

In summary, where the contours allow for it, type designers can create fonts that interpolate between various masters in a multi-dimensional design space. This gives you granular control over your typography, and a great deal of power.

## Axes Definitions

Since the font developers define which axes are available in their fonts, it is essential to check the font's documentation to know what is available. For example, in the Gingham variable font designed by Christoph Koeberlin, there are two axes available, Width and Weight. The Amstelvar variable font does not contain a Slant axis, but does have an axis it calls Grade, plus many more axes.


A Grade axis is interesting as it changes the weight of the font without changing the widths, so line breaks does not change. By playing with a Grade axis, you can avoid being forced to fiddle with changes to Weight axis that effects the overall width, and then changes to the Width axis that effect the overall weight. This is possible because the Amstelvar default style has been deconstructed in the 4 fundamental aspects of form: black or positive shapes, white or negative shapes, and the x and y dimensions. These 4 aspects can be mixed to form the other styles, such as Width and Weight, in the way that primary colors can be mixed to create any other color.

### Amstelvar Alpha

AmstelvarAlpha-Variations.ttf is an [exploratory OpenType 1.8 font](#) made with a combination of Font Bureau's python font tools and the fontmake tool developed by Google.

“The quick brown fox jumps over the lazy dog.”

Grade axis being changed on the fly.

You can view the working example and source code for the above sample [here](#) .

The five registered axes plus Grade have 4-character tags that are used to set their values in CSS:

### Axis names and CSS values

Weight	<b>wght</b>
Width	<b>wdth</b>
Slant	<b>slnt</b>
Optical Size	<b>opsz</b>
Italics	<b>ital</b>
Grade	<b>GRAD</b>

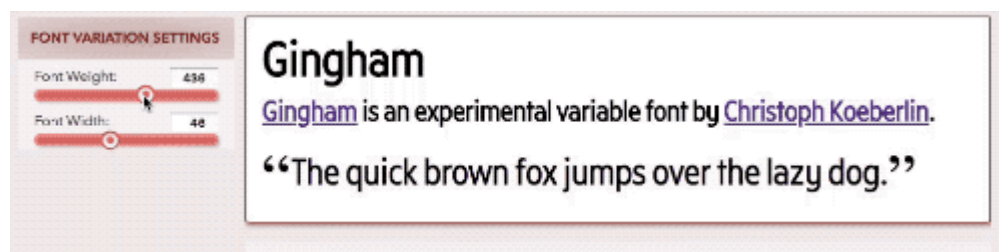
In order to add a variable font first we must link it, as with any custom font;

```
@font-face {  
  font-family: 'AmstelvarAlpha';  
  src: url('../fonts/AmstelvarAlpha-VF.ttf');  
}
```




The way we define an axis value is by using the CSS property **font-variations** which has a series of values that pair the axis tag with an instance location:

```
#font-amstelvar {  
  font-family: 'AmstelvarAlpha';  
  font-variation-settings: 'wdth' 400, 'wght' 98;  
}
```



In this example you can see the Weight and Width axes being changed on the fly.

You can view the working example and source code for the above sample [here](#) .

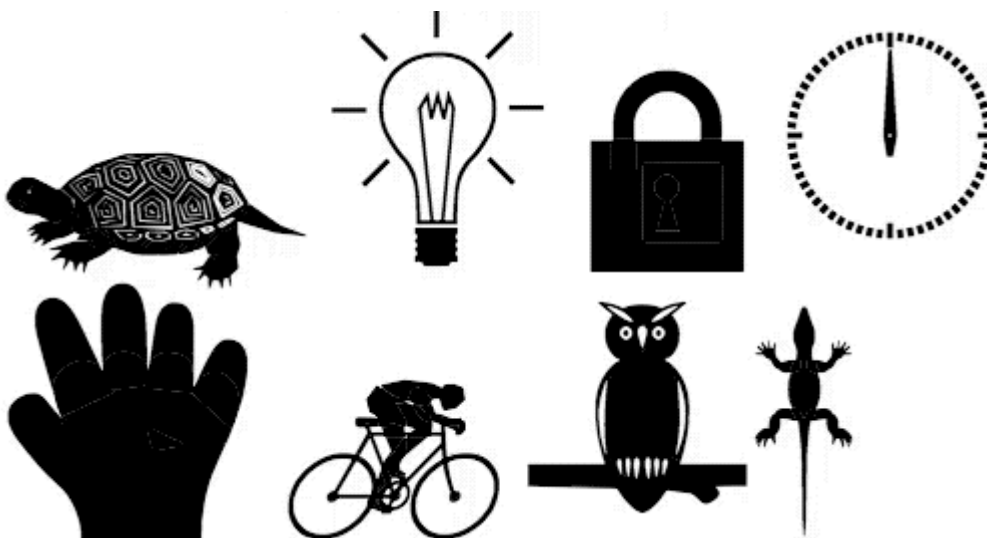
## Responsibility of the typesetter

Setting the axes values comes down to personal taste and applying typographic best practices. The danger with any new technology is possible misuse, and settings that are overly artistic or exploratory could also decrease legibility of the actual text. For titles, exploring different axes to create great artistic designs are exciting, but for body copy this risks making the text illegible.




One great example of artistic expression is shown above, an exploration of the typeface [Decovar](#) [\[1\]](#) by Mandy Michael.

You can view the working example and source code for the above sample [here](#) [\[2\]](#).



There is also a possibility to explore animating characters with variable fonts. Above is an example of different axes being used with the typeface Zycon. See the live [animation example on Axis Praxis](#) [\[3\]](#).


## Variable fonts performance gains

OpenType variable fonts allow us to store multiple variations of a type family into a single font file. [Monotype](#)  ran an experiment by combining 12 input fonts to generate eight weights, across three widths, across both the Italic and Roman styles. Storing 48 individual fonts in a single variable font file meant a **88% reduction in file size**.

On the flip side, if you are animating the font between settings, this may cause the browser performance issues. Learn more about this on [Surma's Supercharged](#).

With variable fonts, app and website makers can offer really rich typography experiences that express each brand, without the previous bandwidth and latency costs. However, if you are using a single font such as Roboto Regular and nothing else, you might see a net gain in font size if you were to switch to a variable font with many axes. As always, it depends on your use-case.

## Fallbacks and browser support

Current support is limited, but variable fonts will work today out of the box in Chrome and Safari, with support coming soon to Edge 17 and Firefox. See [caniuse.com](#)  for more details.

It is possible to use @supports in you CSS to create a viable fallback:




```
@supports (font-variations-settings: 'wdth' 200) {  
  @font-face {  
    /* https://github.com/TypeNetwork/Amstelvar */  
    font-family: AmstelvarAlpha;  
    src: url('../fonts/AmstelvarAlpha-VF.ttf');  
    font-weight: normal;  
    font-style: normal;  
  }  
  
  #font-amstelvar {  
    font-family: AmstelvarAlpha;  
    font-variation-settings: 'wdth' 400, 'wght' 98;  
  }  
}
```



## Thanks

This article would have only been made possible with the help of the following people:



- David Berlow, type designer and typographer at [Font Bureau](#) 
- Laurence Penney, developer of [axis-praxis.org](#) 
- [Mandy Michael](#) 
- Dave Crossland, Program Manager, Google Fonts

---

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated July 2, 2018.*