Variable Fonts in Chrome

Webengines Hackfest, Igalia, A Coruña

Behdad Esfahbod behdad@google.com
Dominik Röttsches drott@google.com

Demos

- Responsive Web Typography
- Font Playground
- Underware’s HOI

Variable Fonts in CSS Level 4 Fonts

font-weight, font-stretch, font-style *before*

3.2. Font weight: the font-weight property

Name:	***font-weight***
Value:	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	numeric weight value (see description)
Animatable:	as font weight

font-weight, font-stretch, font-style *variable*

§ 2.2. Font weight: the 'font-weight' property

Name:	***'font-weight'***
Value:	<font-weight-absolute> | bolder | lighter
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	n/a
Media:	visual
Computed value:	numeric weight value (see description)
Canonical order:	per grammar
Animatable:	As <number>

The 'font-weight' property specifies the weight of glyphs in the font, their degree of blackness or stroke thickness.

This property accepts values of the following:

<font-weight-absolute> = [normal | bold | <number>]

Values have the following meanings:

¶ ***'<number>'***

These values form an ordered sequence, where each number indicates a weight that is at least as dark as its predecessor. Only values greater than or equal to 1, and less than or equal to 1000, are valid, and all other values are treated as parse errors. Certain numeric values correspond to the commonly used weight names below (Note that a font might internally provide its own mappings, but those mappings within the font are disregarded):

Ranges in `@font-face`

```
@font-face {
    font-family: Roboto;
    font-weight: 700;   /* or: 400, 600, 900,... */
    font-style: normal; /* or: italic, oblique */
    font-stretch: condensed; /* or: expanded, ultra-expanded */
}
```

Ranges in `@font-face`

```
@font-face {
    font-family: Roboto;
    font-weight: 400 700;
    font-style: 10deg 20deg;
    font-stretch: 50% 200%;
}
```

New Font Style Matching Algorithm

- https://drafts.csswg.org/css-fonts-4/#font-style-matching
- Previously, for a font request:
  - Match `font-stretch`, `font-style`, `font-weight`
    by traversing keyword values, find closest keyword
- New definition: Search for numerically nearest value
  - As defined by `@font-face` and
  - Within the range that the variable font allows

font-optical-sizing

§ 8.1. Optical sizing control: the ‘font-optical-sizing’ property

Name:	***‘font-optical-sizing’***
Value:	auto | none
Initial:	auto
Applies to:	all elements
Inherited:	yes
Percentages:	n/a
Media:	visual
Computed value:	as specified
Canonical order:	per grammar
Animatable:	no

font-variation-settings

- Similar to font-feature-settings
- Sequence of 4 character axis name plus

```
font-variation-settings: ‘wght’ 700, ‘UPWD’ 200;
```

Variable Fonts in Blink

New CSS Font Matching Algorithm

- Implements `font-stretch`, `font-style`, `font-weight` matching based on numbers, not based on keywords
- `FontTraits` replaced with `FontSelectionRequest`
  - Now storing three `FontSelectionValues` (numerical values for stretch, style weight)
- `FontSelectionCapabilities` are storing what the `@font-face` definition provides

Example: Font Style Matching - Before

```
@font-face {
Font-family: Roboto;
Src: url(Roboto-light.otf);
Font-weight: 200;
}
```

```
@font-face {
Font-family: Roboto;
Src: url(Roboto-regular.otf);
Font-weight: 400;
}
```

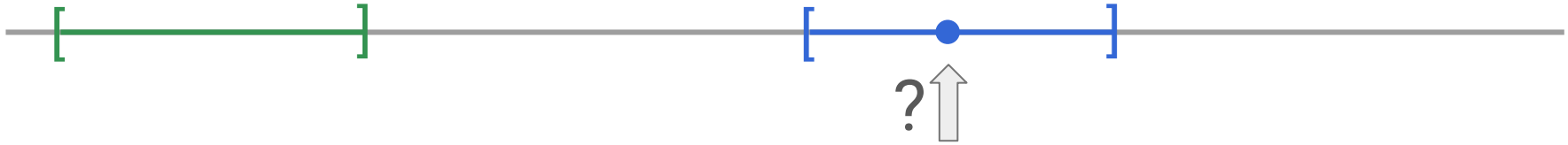
```
@font-face {
Font-family: Roboto;
Src: url(Roboto-bold.otf);
Font-weight: 700;
}
```


```
<div style="font-weight: 600;">Bold text</div>
```

Example: Font Style Matching - New

```
@font-face {
Font-family: Roboto;
Src: url(Roboto-lighter.otf);
Font-weight: 100 300;
}
```

```
@font-face {
Font-family: Roboto;
Src: url(Roboto-bolder.otf);
Font-weight: 500 700;
}
```




```
<div style="font-weight: 600;">Bold text</div>
```

Rasterizing Variable Fonts

- Rasterization of variable fonts is controlled by axis parameters
- Passing variation axes parameters to Skia

```
SkFontMgr::FontParameters::Axis weight_axis = {
        SkSetFourByteTag('w', 'g', 'h', 't'),

SkFloatToScalar(selection_capabilities.weight.clampToRange(
            selection_request.weight))};
```

The Cross-platform Challenge

- Skia uses platform specific font rasterization engines
- Only FreeType, CoreText on Mac 10.12 support, newer Windows 10 support rasterizing

?



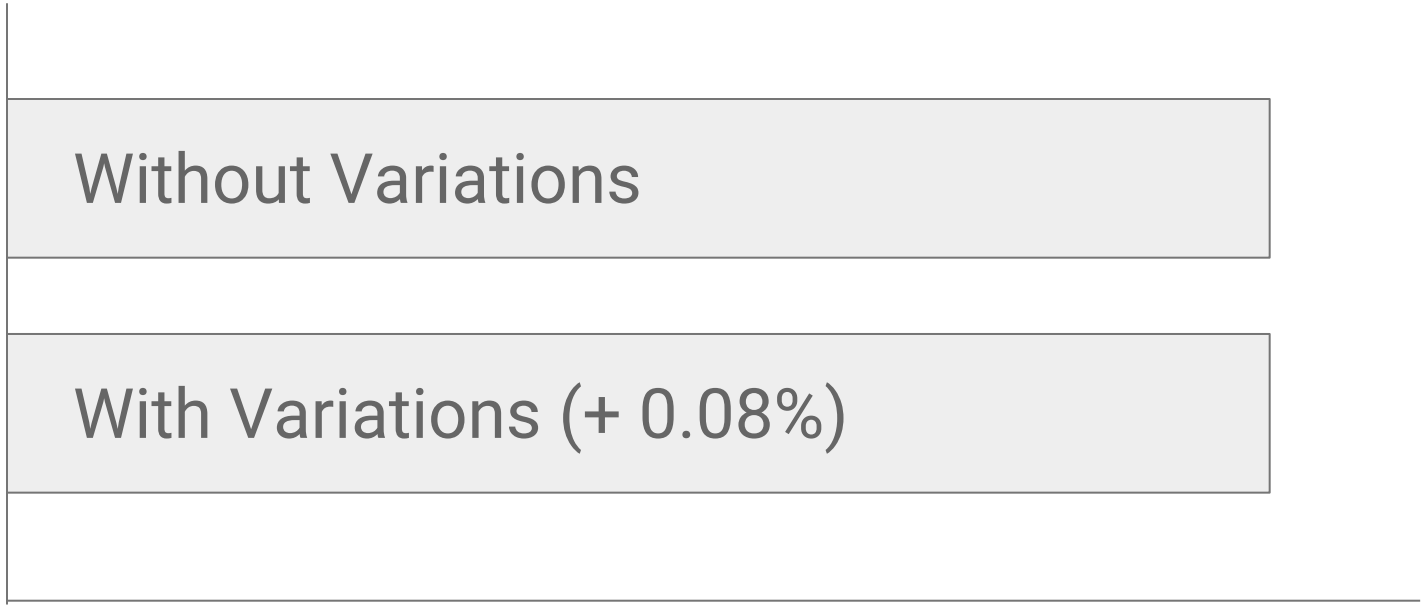


OS	ChromeOS	Linux	Android	Windows	Mac OS
Font Rasterizing for existing fonts	FreeType	FreeType	FreeType	DirectWrite	CoreText
Variable Fonts	FreeType	FreeType	FreeType	FreeType	CoreText FreeType

Font-format specific web font instantiation

- Is this web font variable? = Does it have an `fvar` table?
  - If yes, do we have platform support?
    - If yes, use the platform rasterizer
    - If not, use built-in FreeType!
  - If it is not variable, keep using the existing platform rasterizer

Windows Chrome Binary Size



Shipping FreeType on all platforms

- FreeType harmonized and unified between PDFium and Blink
- `third_party/freetype2` and `third_party/freetype-android` unified
- 3 different checkouts for building Chrome reduced to one

Chromium Blog

News and developments from the open source browser project

Chrome 62 Beta: Network Quality Estimator API, OpenType variable fonts, and media capture from DOM elements

Wednesday, September 20, 2017

Unless otherwise noted, changes described below apply to the newest Chrome Beta channel release for Android, Chrome OS, Linux, Mac, and Windows.

Network Quality Estimator API

The Network Infomation API has been available in previous versions of Chrome, but has only provided theoretical network speeds given the type of a user's connection. In this release, the API has been expanded to provide developers with network performance metrics as experienced by the client. Using the API, a developer can inspect the current expected round trip time and throughput and be notified of performance changes. To simplify application logic, the API also summarizes measured network performance as the cellular connection type (e.g. `2G`) most similar to it, even if the actual connection is WiFi or Ethernet.

Using these network quality signals, developers can tailor content to network constraints. For example, on very slow connections, developers can serve a simplified version of the page to improve page load times. These signals will also soon be available as HTTP request headers and enabled via Client Hints.

OpenType Variable Fonts

OpenType Font Variations bring new typographic capabilities to the web. Previously, one font file contained just a single instance of a font family, including only one weight (Regular, Bold, Black…) or one stretch (Normal, Condensed, Expanded…).

Figure: Animated Amstelvar and Decovar variable font examples

With variable fonts, responsive design on the web now extends to typography. OpenType Variations provide a continuous spectrum of stylistic variations while saving space and bandwidth, since they all load from a single compact font file. Stretch, style, and weight can be adjusted using the respective updated CSS properties which now allow numeric values. Fine tuning of variation axis parameters, such as weight or width, is possible using the `font-variation-settings` CSS property.

https://blog.chromium.org/2017/09/chrome-62-beta-network-quality.html

New applications of Hybrid Font Stack

CFF2

- Adobe CFF2 format, alternative to TrueType contours format
- Adobe Variable Font Prototype exists as CFF2 version

CBDT / CBLC

- Color font format
- Noto Color Emoji

SBIX

- Color font format
- Apple Color Emoji

COLR/CPAL

- Color Font Format
- Twemoji, COLR/CPAL example font, Mozilla’s default emoji font

Summary

- Hybrid Font Stack without increasing binary size
- Cross Platform Support
- Reaping the benefits: Additional format support for color and CFF2 fonts

Chromacheck

- https://pixelambacht.nl/chromacheck/