The fontspec package

WILL ROBERTSON and KHALED HOSNY
will.robertson@latex-project.org
2012/05/06 v2.2b

Contents

1 History 3
2 Introduction 3
  2.1 About this manual . 3
  2.2 Acknowledgements . 4
3 Package loading and options 4
  3.1 Maths fonts adjustments 5
  3.2 Configuration . . . 5
  3.3 Warnings . . . . 5
I General font selection 5
4 Font selection 6
  4.1 By font name . . . . 6
  4.2 By file name . . . . 6
5 Default font families 8
6 New commands to select font families 8
  6.1 More control over font shape selection . . . 10
  6.2 Math(s) fonts . . . . . 11
  6.3 Miscellaneous font selecting details . . . . 12
7 Selecting font features 12
  7.1 Default settings . . . 13
  7.2 Changing the currently selected features . . . 13
  7.3 Priority of feature selection . . . . . . 13
8 Font independent options 16
  8.1 Colour . . . . . . 16
  8.2 Scale . . . . . 17
  8.3 Interword . . . . 17
  8.4 Post-punctuation space 18
  8.5 The hyphenation character . . . . . 18
  8.6 Optical font sizes . . 19
II OpenType 20
9 Introduction 20
  9.1 How to select font features . . . . . . . . 21
10 Complete listing of OpenType font features 21
  10.1 Ligatures . . . . . . 21
  10.2 Letters . . . . . . 23
  10.3 Numbers . . . . . . 24
  10.4 Contextuals . . . . . 25
  10.5 Vertical Position . . . 25
  10.6 Fractions . . . . . . 26
  10.7 Stylistic Set variations . . . 26
  10.8 Character Variants . . . . 27
  10.9 Alternates . . . . . . 27
  10.10 Style . . . . . . . . . 29
  10.11 Diacritics . . . . . . . 30
  10.12 Kerning . . . . . . . 30
  10.13 Font transformations . 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.14</td>
<td>Annotation</td>
<td>32</td>
</tr>
<tr>
<td>10.15</td>
<td>CJK shape</td>
<td>32</td>
</tr>
<tr>
<td>10.16</td>
<td>Character width</td>
<td>32</td>
</tr>
<tr>
<td>10.17</td>
<td>Vertical typesetting</td>
<td>35</td>
</tr>
<tr>
<td>10.18</td>
<td>OpenType scripts and languages</td>
<td>35</td>
</tr>
<tr>
<td>III</td>
<td>Lua\TeX-only font features</td>
<td>38</td>
</tr>
<tr>
<td>11</td>
<td>OpenType font feature files</td>
<td>38</td>
</tr>
<tr>
<td>IV</td>
<td>Fonts and features with X\TeX</td>
<td>38</td>
</tr>
<tr>
<td>12</td>
<td>X\TeX-only font features</td>
<td>39</td>
</tr>
<tr>
<td>12.1</td>
<td>Mapping</td>
<td>39</td>
</tr>
<tr>
<td>12.2</td>
<td>Letter spacing</td>
<td>39</td>
</tr>
<tr>
<td>12.3</td>
<td>Different font technologies: AAT and ICU</td>
<td>40</td>
</tr>
<tr>
<td>12.4</td>
<td>Optical font sizes</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>Mac OS X’s AAT fonts</td>
<td>41</td>
</tr>
<tr>
<td>13.1</td>
<td>Ligatures</td>
<td>41</td>
</tr>
<tr>
<td>13.2</td>
<td>Letters</td>
<td>41</td>
</tr>
<tr>
<td>13.3</td>
<td>Numbers</td>
<td>41</td>
</tr>
<tr>
<td>13.4</td>
<td>Contextuals</td>
<td>42</td>
</tr>
<tr>
<td>13.5</td>
<td>Vertical position</td>
<td>42</td>
</tr>
<tr>
<td>13.6</td>
<td>Fractions</td>
<td>43</td>
</tr>
<tr>
<td>13.7</td>
<td>Variants</td>
<td>43</td>
</tr>
<tr>
<td>13.8</td>
<td>Alternates</td>
<td>43</td>
</tr>
<tr>
<td>13.9</td>
<td>Style</td>
<td>44</td>
</tr>
<tr>
<td>13.10</td>
<td>CJK shape</td>
<td>44</td>
</tr>
<tr>
<td>13.11</td>
<td>Character width</td>
<td>44</td>
</tr>
<tr>
<td>13.12</td>
<td>Vertical typesetting</td>
<td>44</td>
</tr>
<tr>
<td>13.13</td>
<td>Diacritics</td>
<td>45</td>
</tr>
<tr>
<td>13.14</td>
<td>Annotation</td>
<td>45</td>
</tr>
<tr>
<td>14</td>
<td>AAT &amp; Multiple Master font axes</td>
<td>45</td>
</tr>
<tr>
<td>V</td>
<td>Programming interface</td>
<td>46</td>
</tr>
<tr>
<td>15</td>
<td>Defining new features</td>
<td>46</td>
</tr>
<tr>
<td>16</td>
<td>Going behind fontspec’s back</td>
<td>47</td>
</tr>
<tr>
<td>17</td>
<td>Renaming existing features &amp; options</td>
<td>48</td>
</tr>
<tr>
<td>18</td>
<td>Programming details</td>
<td>48</td>
</tr>
<tr>
<td>VI</td>
<td>The patching/improvement of \LaTeX,2ε and other packages</td>
<td>50</td>
</tr>
<tr>
<td>19</td>
<td>Inner emphasis</td>
<td>50</td>
</tr>
<tr>
<td>20</td>
<td>Unicode footnote symbols</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>Verbatim</td>
<td>50</td>
</tr>
<tr>
<td>22</td>
<td>Discretionary hyphenation: \textbackslash-</td>
<td>50</td>
</tr>
<tr>
<td>23</td>
<td>Commands for old-style and lining numbers</td>
<td>51</td>
</tr>
<tr>
<td>VII</td>
<td>\fontspec.sty and friends</td>
<td>52</td>
</tr>
<tr>
<td>24</td>
<td>‘Header’ code</td>
<td>52</td>
</tr>
<tr>
<td>24.1</td>
<td>expl3 tools</td>
<td>52</td>
</tr>
<tr>
<td>24.2</td>
<td>Bits and pieces</td>
<td>52</td>
</tr>
<tr>
<td>24.3</td>
<td>Error/warning/info messages</td>
<td>53</td>
</tr>
<tr>
<td>24.4</td>
<td>Option processing</td>
<td>57</td>
</tr>
<tr>
<td>24.5</td>
<td>Packages</td>
<td>57</td>
</tr>
<tr>
<td>25</td>
<td>The main package code</td>
<td>58</td>
</tr>
<tr>
<td>25.1</td>
<td>Encodings</td>
<td>58</td>
</tr>
<tr>
<td>25.2</td>
<td>User commands</td>
<td>58</td>
</tr>
<tr>
<td>25.3</td>
<td>Programmer’s interface</td>
<td>64</td>
</tr>
<tr>
<td>25.4</td>
<td>expl3 interface for font loading</td>
<td>68</td>
</tr>
<tr>
<td>25.5</td>
<td>Internal macros</td>
<td>69</td>
</tr>
<tr>
<td>25.6</td>
<td>keyval definitions</td>
<td>85</td>
</tr>
<tr>
<td>25.7</td>
<td>Italic small caps</td>
<td>108</td>
</tr>
<tr>
<td>25.8</td>
<td>Selecting maths fonts</td>
<td>109</td>
</tr>
<tr>
<td>25.9</td>
<td>Finishing up</td>
<td>113</td>
</tr>
<tr>
<td>25.10</td>
<td>Compatibility</td>
<td>113</td>
</tr>
</tbody>
</table>
1 History

This package began life as a \LaTeX{} interface to select system-installed Mac OS X fonts in Jonathan Kew’s \XeTeX{}, the first widely-used Unicode extension to \TeX{}. Over time, \XeTeX{} was extended to support OpenType fonts and then was ported into a cross-platform program to run also on Windows and Linux.

More recently, \LuaTeX{} is fast becoming the \TeX{} engine of the day; it supports Unicode encodings and OpenType fonts and opens up the internals of \TeX{} via the Lua programming language. Hans Hagen’s \ConTeXt{} Mk. IV is a re-write of his powerful typesetting system, taking full advantage of \LuaTeX{}’s features including font support; a kernel of his work in this area has been extracted to be useful for other \TeX{} macro systems as well, and this has enabled \fontspec{} to be adapted for \LaTeX{} when run with the \LuaTeX{} engine. Elie Roux and Khaled Hosny have been instrumental and invaluable with this development work.

2 Introduction

The \fontspec{} package allows users of either \XeTeX{} or \LuaTeX{} to load OpenType fonts in a \LaTeX{} document. No font installation is necessary, and font features can be selected and used as desired throughout the document.

Without \fontspec{}, it is necessary to write cumbersome font definition files for \LaTeX{}, since \LaTeX{}’s font selection scheme (known as the ‘nfss’) has a lot going on behind the scenes to allow easy commands like \texttt{\emph{}} or \texttt{\bfseries{}}. With an uncountable number of fonts now available for use, however, it becomes less desirable to have to write these font definition (.fd) files for every font one wishes to use.

Because \fontspec{} is designed to work in a variety of modes, this user documentation is split into separate sections that are designed to be relatively independent. Nonetheless, the basic functionality all behaves in the same way, so previous users of \fontspec{} under \XeTeX{} should have little or no difficulty switching over to \LuaTeX{}.

This manual can get rather in-depth, as there are a lot of details to cover. See the example documents \texttt{fontspec-xetex.tex} and \texttt{fontspec-luatex.tex} for a complete minimal example with each engine.

2.1 About this manual

This document is typeset with \pdflatex{} using pre-compiled examples that have been generated by either \XeTeX{} or \LuaTeX{}. You may regenerate the examples.
by removing the doc-files/ subdirectory and typesetting the manual with the following invocation:

    pdflatex -shell-escape fontspec.dtx

Note that many of the examples use fonts that are not included in \TeX{} Live or MiKTeX, and some of them are non-free fonts that must be purchased.

I’d like to reduce the number of non-free fonts used in this manual. If you know any freely available fonts that could be used as alternative to any of the fonts in this document, please suggest them to me. Finally, if any aspect of the documentation is unclear or you would like to suggest more examples that could be made, get in touch. (Contributions especially welcome.)

2.2 Acknowledgements

This package couldn’t be possible without the early and continued support the author of X\TeX{}, Jonathan Kew. When I started this package, he steered me many times in the right direction.

I’ve had great feedback over the years on feature requests, documentation queries, bug reports, font suggestions, and so on from lots of people all around the world. Many thanks to you all.

Thanks to David Perry and Markus Böhning for numerous documentation improvements and David Perry again for contributing the text for one of the sections of this manual.

Special thanks to Khaled Hosny, who had been the driving force behind the support for Lua\TeX{}, ultimately leading to version 2.0 of the package.

3 Package loading and options

For basic use, no package options are required:

    \usepackage{fontspec}

\texttt{xunicode}  Ross Moore’s \texttt{xunicode} package is now automatically loaded for users of both X\TeX{} and Lua\TeX{}. This package provides backwards compatibility with \texttt{\LaTeX{}}’s methods for accessing extra characters and accents (for example, ∇, $, \textbullet{}, "u, and so on), plus many more Unicode characters. \textbf{Warning}: introduced in v2.1, this is a backwards incompatible change to previous versions of fontspec. This change was necessary in order to provide consistent support for users of X\TeX{} and Lua\TeX{}. I’m not aware of any issues this may cause but please let me know if you run into problems.

\textbf{X\TeX{} users only}  The \texttt{xlatex} package adds some minor extra features to X\TeX{}, including, via the metalogo package, the \texttt{\LaTeX} macro to typeset the X\TeX{} logo. While this package was previously recommended, it serves a much smaller rôle nowadays and generally will not be required.
LuaTeX users only  In order to load fonts by their name rather than by their filename (e.g., 'Latin Modern Roman' instead of 'ec-lmr10'), you may need to run the script \texttt{mkluatexfontdb}, which is distributed with the luatexload package. Note that if you do not execute this script beforehand, the first time you attempt to typeset the process will pause for (up to) several minutes. (But only the first time.) Please see the luatexload documentation for more information.

babel  The babel package is not really supported! Especially Vietnamese, Greek, and Hebrew at least might not work correctly, as far as I can tell. There’s a better chance with Cyrillic and Latin-based languages, however—fontspec ensures at least that fonts should load correctly, but hyphenation and other matters aren’t guaranteed. Under Xe\TeX, the polyglossia package is recommended instead as a modern replacement for babel.

3.1 Maths fonts adjustments

By default, fontspec adjusts \LaTeX’s default maths setup in order to maintain the correct Computer Modern symbols when the roman font changes. However, it will attempt to avoid doing this if another maths font package is loaded (such as mathpazo or the unicode-math package).

If you find that fontspec is incorrectly changing the maths font when it should be leaving well enough alone, apply the [no-math] package option to manually suppress its maths font.

3.2 Configuration

If you wish to customise any part of the fontspec interface (see later in this manual, Section 15 on page 46 and Section 17), this should be done by creating your own \texttt{fontspec.cfg} file, which will be automatically loaded if it is found by Xe\TeX or Lua\TeX. Either place it in the same folder as the main document for isolated cases, or in a location that Xe\TeX or Lua\TeX searches by default; e.g. in Mac\TeX: 
\texttt{~/Library/texmf/tex/latex/}. The package option \texttt{[no-config]} will suppress this behaviour under all circumstances.

3.3 Warnings

This package can give many warnings that can be harmless if you know what you’re doing. Use the [quiet] package option to write these warnings to the transcript (.log) file instead.

Use the [silent] package option to completely suppress these warnings if you don’t even want the .log file cluttered up.

Part I

General font selection

This section concerns the variety of commands that can be used to select fonts.
These are the main font-selecting commands of this package. The \fontspec command selects a font for one-time use; all others should be used to define the standard fonts used in a document. They will be described later in this section.

The font features argument accepts comma separated ⟨font feature⟩=⟨option⟩ lists; these are described in later:

- For general font features, see Section 8 on page 16
- For OpenType fonts, see Part II on page 20
- For \XeTeX-only general font features, see Part IV on page 39
- For Lua\TeX-only general font features, see Part III on page 38
- For features for \aat fonts in \XeTeX, see Section 13 on page 41

4 Font selection

In both Lua\TeX and \XeTeX, fonts can be selected either by ‘font name’ or by ‘file name’.

4.1 By font name

Fonts known to Lua\TeX or \XeTeX may be loaded by their names. ‘Known to’ in this case generally means ‘exists in a standard fonts location’ such as `~/Library/Fonts on Mac OS X, or `C:\Windows\Fonts on Windows.

The simplest example might be something like

\fontspec [... ]{Cambria}

in which the bold and italic fonts will be found automatically (if they exist) and are immediately accessible with the usual \textit and \textbf commands.

TODO: add explanation for how to find out what the ‘font name’ is.

4.2 By file name

\XeTeX and Lua\TeX also allow fonts to be loaded by file name instead of font name. When you have a very large collection of fonts, you will sometimes not wish to have them all installed in your system’s font directories. In this case, it is more convenient to load them from a different location on your disk. This technique is also necessary in \XeTeX when loading OpenType fonts that are present within your \TeX distribution, such as `/usr/local/texlive/2010/texmf-dist/fonts/opentype/public. Fonts in such locations are visible to \XeTeX but cannot be loaded by font name, only file name; Lua\TeX does not have this restriction.
When selecting fonts by file name, any font that can be found in the default search paths may be used directly (including in the current directory) without having to explicitly define the location of the font file on disk.

Xe\LaTeX & Mac users only: Note that Xe\LaTeX can only select fonts in this way with the xdvipdfmx driver, but Xe\LaTeX with the xdvi2pdf driver can only select system-installed fonts by font name and not file name. The xdvipdfmx driver is default for Xe\LaTeX, so this is only a problem if you wish to explicitly use the xdvi2pdf driver.

Fonts selected by filename must include bold and italic variants explicitly.

\fontspec
[ BoldFont = texgyrepagella-bold.otf,
  ItalicFont = texgyrepagella-italic.otf,
  BoldItalicFont = texgyrepagella-bolditalic.otf ]
{texgyrepagella-regular.otf}

\fontspec
[ Extension = .otf,
  BoldFont = texgyrepagella-bold,
  ... ]
{texgyrepagella-regular}

If desired, an abbreviation can be applied to the font names based on the mandatory `font name' argument:

\fontspec
[ Extension = .otf,
  UprightFont = *-regular,
  BoldFont = *-bold,
  ... ]
{texgyrepagella}

In this case `texgyrepagella' is no longer the name of an actual font, but is used to construct the font names for each shape; the * is replaced by `texgyrepagella'. Note in this case that UprightFont is required for constructing the font name of the normal font to use.

To load a font that is not in one of the default search paths, its location in the filesystem must be specified with the Path feature:

\fontspec
[ Path = /Users/will/Fonts/,
  UprightFont = *-regular,
  BoldFont = *-bold,
  ... ]
{texgyrepagella}
Example 1: Loading the default, sans serif, and monospaced fonts.

\setmainfont{TeX Gyre Bonum}
\setsansfont[Scale=MatchLowercase]{Latin Modern Sans}
\setmonofont[Scale=MatchLowercase]{Inconsolata}

\rmfamily Pack my box with five dozen liquor jugs \par
\sffamily Pack my box with five dozen liquor jugs \par
\ttfamily Pack my box with five dozen liquor jugs

Note that X\TeX{} and Lua\TeX{} are able to load the font without giving an extension, but fontspec must know to search for the file; this can be indicated by declaring the font exists in an ‘ExternalLocation’:

\fontspec
[ Externallocation ,
  BoldFont = texgyrepagella-bold ,
  \ldots ]
{texgyrepagella-regular}

To be honest, Path and Externallocation are actually the same feature with different names. The former can be given without an argument and the latter can be given with one; the different names are just for clarity.

5 Default font families

\setmainfont [{⟨font features⟩}] {⟨font name⟩}
\setsansfont [{⟨font features⟩}] {⟨font name⟩}
\setmonofont [{⟨font features⟩}] {⟨font name⟩}

These commands are used to select the default font families for the entire document. They take the same arguments as \fontspec. See Example 1. Here, the scales of the fonts have been chosen to equalise their lowercase letter heights. The Scale font feature will be discussed further in Section 8 on page 16, including methods for automatic scaling.

6 New commands to select font families

\newfontfamily \{⟨font-switch⟩\} [{⟨font features⟩}] {⟨font name⟩}
\newfontface \{⟨font-switch⟩\} [{⟨font features⟩}] {⟨font name⟩}

For cases when a specific font with a specific feature set is going to be re-used many times in a document, it is inefficient to keep calling \fontspec for every use. While the \fontspec command does not define a new font instance after the first call, the feature options must still be parsed and processed.

\newfontfamily For this reason, new commands can be created for loading a particular font fami-
Example 2: Defining new font families.

\newfontfamily\notefont{Kurier}
\notefont This is a \emph{note}.

Example 3: Defining a single font face.

\newfontface\fancy
[Contextuals={WordInitial,WordFinal}]
\Hoefler Text Italic
\fancy where is all the vegemite
% \emph, \textbf, etc., all don't work

ily with the \newfontfamily command, demonstrated in Example 2. This macro should be used to create commands that would be used in the same way as \rmfamily, for example. If you would like to create a command that only changes the font inside its argument (i.e., the same behaviour as \emph) define it using regular \LaTeX{} commands:

\newcommand\textnote[1]{{\notefont #1}}
\textnote{This is a note.}

Note that the double braces are intentional; the inner pair are used to to delimit the scope of the font change.

Sometimes only a specific font face is desired, without accompanying italic or bold variants being automatically selected. This is common when selecting a fancy italic font, say, that has swash features unavailable in the upright forms. \newfontface is used for this purpose, shown in Example 3, which is repeated in Section 13.4 on page 42.

Comment for advanced users: The commands defined by \newfontface and \newfontfamily include their encoding information, so even if the document is set to use a legacy \TeX{} encoding, such commands will still work correctly. For example,

\documentclass{article}
\usepackage{fontspec}
\newfontfamily\unicodefont{Lucida Grande}
\usepackage{mathpazo}
\usepackage[T1]{fontenc}
\begin{document}
A legacy \TeX{} font. (\unicodefont A unicode font.)
\end{document}
6.1 More control over font shape selection

The automatic bold, italic, and bold italic font selections will not be adequate for the needs of every font: while some fonts mayn’t even have bold or italic shapes, in which case a skilled (or lucky) designer may be able to chose well-matching accompanying shapes from a different font altogether, others can have a range of bold and italic fonts to chose among. The **BoldFont** and **ItalicFont** features are provided for these situations. If only one of these is used, the bold italic font is requested as the default from the new font. See Example 4.

If a bold italic shape is not defined, or you want to specify both custom bold and italic shapes, the **BoldItalicFont** feature is provided.

### 6.1.1 Input shorthands

For those cases that the base font name is repeated, you can replace it with an asterisk. (This has been shown previously in Section 4.2 on page 6.) For example, some space can be saved instead of writing ‘Baskerville SemiBold’:

\fontspec[BoldFont={* SemiBold}]{Baskerville}

As a matter of fact, this feature can also be used for the upright font too:

\fontspec[UprightFont={* SemiBold},
BoldFont={* Bold}]{Baskerville}

### 6.1.2 Small caps and slanted font shapes

For the rare situations where a font family will have slanted and italic shapes, these may be specified separately using the analogous features **SlantedFont** and **BoldSlantedFont**. Without these, however, the LATEX font switches for slanted (**\textsl**, \slshape) will default to the italic shape.
Old-fashioned font families used to distribute their small caps glyphs in separate fonts due to the limitations on the number of glyphs allowed in the PostScript Type 1 format. Such fonts may be used by declaring the `SmallCapsFont` of the family you are specifying:

```
\fontspec[
  SmallCapsFont={Minion MM Small Caps & Oldstyle Figures},
](Minion MM Roman)
```

Roman 123 \ \textsc{Small caps 456}

In fact, you may specify the small caps font for each individual bold and italic shape as in

```
\fontspec[
  UprightFeatures = { SmallCapsFont={ <sc> } },
  BoldFeatures = { SmallCapsFont={ <bf sc> } },
  ItalicFeatures = { SmallCapsFont={ <it sc> } },
  BoldItalicFeatures = { SmallCapsFont={ <bf it sc> } },
](<upright>)
```

Roman 123 \ \textsc{Small caps 456}

For most modern fonts that have small caps as a font feature, this level of control isn’t generally necessary, but you may still occasionally find font families in which the small caps are in a separate font.

All of the bold, italic, and small caps fonts can be loaded with different font features from the main font. See Section 7.4 for details. When an OpenType font is selected for `SmallCapsFont`, the small caps font feature is not automatically enabled. In this case, users should write instead

```
\fontspec[
  SmallCapsFont={...},
  SmallCapsFeatures={Letters=SmallCaps},
](...)
```

6.2 Math(s) fonts

When `\setmainfont`, `\setsansfont` and `\setmonofont` are used in the preamble, they also define the fonts to be used in maths mode inside the `\mathrm`-type commands. This only occurs in the preamble because \LaTeX freezes the maths fonts after this stage of the processing. The fontspec package must also be loaded after any maths font packages (e.g., `euler`) to be successful. (Actually, it is only `euler` that is the problem.\footnote{Speaking of `euler`, if you want to use its \texttt{[mathbf]} option, it won’t work, and you’ll need to put this after fontspec is loaded instead: \AtBeginDocument{\DeclareMathAlphabet\mathbf{U}{eur}{b}{n}}})

Note that fontspec will not change the font for general mathematics; only the upright and bold shapes will be affected. To change the font used for the mathematical symbols, see either the mathspec package or the unicode-math package.

Note that you may find that loading some maths packages won’t be as smooth as you expect since fontspec (and \LaTeX in general) breaks many of the assumptions
of \TeX as to where maths characters and accents can be found. Contact me if you have troubles, but I can't guarantee to be able to fix any incompatibilities. The Lucida and Euler maths fonts should be fine; for all others keep an eye out for problems.

\begin{verbatim}
\setmathrm [(font features)]{(font name)}
\setmathsf [(font features)]{(font name)}
\setmathtt [(font features)]{(font name)}
\setboldmathrm [(font features)]{(font name)}
\end{verbatim}

However, the default text fonts may not necessarily be the ones you wish to use when typesetting maths (especially with the use of fancy ligatures and so on). For this reason, you may optionally use the commands above (in the same way as our other \fontspec-like commands) to explicitly state which fonts to use inside such commands as \mathrm. Additionally, the \setboldmathrm command allows you define the font used for \mathrm when in bold maths mode (which is activated with, among others, \boldmath).

For example, if you were using Optima with the Euler maths font, you might have this in your preamble:

\begin{verbatim}
usepackage{mathpazo}
usepackage{fontspec,xunicode}
setmainfont{Optima}
setmathrm{Optima}
setboldmathrm{Optima ExtraBlack}[(font features)]{Optima Bold}
\end{verbatim}

6.3 Miscellaneous font selecting details

Spaces \fontspec and \addfontfeatures ignore trailing spaces as if it were a ‘naked’ control sequence; e.g., ‘M. \fontspec{...} N’ and ‘M. \fontspec{...}N’ are the same.

Italic small caps Note that this package redefines the \itshape and \scshape commands in order to allow them to select italic small caps in conjunction.

Emphasis and nested emphasis You may specify the behaviour of the \emph command by setting the \emshape command. E.g., for bold emphasis:

\begin{verbatim}
\renewcommand{\emshape}{\bfseries}
\end{verbatim}

Nested emphasis is controlled by the \eminnershape command. For example, for \emph{\emph{...}} to produce small caps:

\begin{verbatim}
\renewcommand{\eminnershape}{\scshape}
\end{verbatim}

7 Selecting font features

The commands discussed so far such as \fontspec each take an optional argument for accessing the font features of the requested font. Commands are provided to set default features to be applied for all fonts, and even to change the features that a font is presently loaded with. Different font shapes can be loaded with separate
features, and different features can even be selected for different sizes that the font appears in. This section discusses these options.

7.1 Default settings

\defaultfontfeatures{(font features)}

It is desirable to define options that are applied to every subsequent font selection command: a default feature set, so to speak. This may be defined with the \defaultfontfeatures command, shown in Example 5. New calls of \defaultfontfeatures overwrite previous ones.

7.2 Changing the currently selected features

\addfontfeatures{(font features)}

This command allows font features to be changed without knowing what features are currently selected or even what font is being used. A good example of this could be to add a hook to all tabular material to use monospaced numbers, as shown in Example 6.

\addfontfeature

This command may also be executed under the alias \addfontfeature.

7.3 Priority of feature selection

Features defined with \addfontfeatures override features specified by \fontspec, which in turn override features specified by \defaultfontfeatures. If in doubt, whenever a new font is chosen for the first time, an entry is made in the transcript (.log) file displaying the font name and the features requested.
Example 6: A demonstration of the `\addfontfeatures` command.

```
\fontspec[Numbers={Proportional,OldStyle}]
{TeX Gyre Adventor}
\textquote{In 1842, 999 people sailed 97 miles in 13 boats. In 1923, 111 people sailed 54 miles in 56 boats.} \bigskip
\addfontfeatures{Numbers={Monospaced,Lining}}
\begin{tabular}{@{} cccc @{}}
  Year & People & Miles & Boats \\
 1842 & 999 & 75 & 13 \\
 1923 & 111 & 54 & 56
\end{tabular}
```

"In 1842, 999 people sailed 97 miles in 13 boats. In 1923, 111 people sailed 54 miles in 56 boats."

<table>
<thead>
<tr>
<th>Year</th>
<th>People</th>
<th>Miles</th>
<th>Boats</th>
</tr>
</thead>
<tbody>
<tr>
<td>1842</td>
<td>999</td>
<td>75</td>
<td>13</td>
</tr>
<tr>
<td>1923</td>
<td>111</td>
<td>54</td>
<td>56</td>
</tr>
</tbody>
</table>

Example 7: Features for, say, just italics.

```
\textit{Attention All Martini Drinkers} \textit{Attention All Martini Drinkers}
\addfontfeature{ItalicFeatures={Alternate = 1}}
\textit{Attention All Martini Drinkers} \textit{Attention All Martini Drinkers}
```

7.4 Different features for different font shapes

```
\begin{verbatim}
BoldFeatures\{\textit{features}\}
ItalicFeatures\{\textit{features}\}
BoldItalicFeatures\{\textit{features}\}
SlantedFeatures\{\textit{features}\}
BoldSlantedFeatures\{\textit{features}\}
SmallCapsFeatures\{\textit{features}\}
\end{verbatim}
```

It is entirely possible that separate fonts in a family will require separate options; e.g., Hoefler Text Italic contains various swash feature options that are completely unavailable in the upright shapes.

The font features defined at the top level of the optional `\fontspec` argument are applied to all shapes of the family. Using `Upright-`, `SmallCaps-`, `Bold-`, `Italic-`, and `BoldItalicFeatures`, separate font features may be defined to their respective shapes in addition to, and with precedence over, the ‘global’ font features. See Example 7.

Combined with the options for selecting arbitrary fonts for the different shapes, these separate feature options allow the selection of arbitrary weights in the Skia typeface, as shown in Example 8.

Note that because most fonts include their small caps glyphs within the main font, features specified with `SmallCapsFeatures` are applied in addition to any other shape-specific features as defined above, and hence `SmallCapsFeatures` can
Example 8: Multiple Master–like features in AAT fonts.

\begin{verbatim}
\fontspec[BoldFont={Skia},
            BoldFeatures={Weight=2}]{Skia}
\fontspec[BoldFont={Skia}]{Skia} \bfseries Skia 'Bold'
\end{verbatim}

Example 9: An example of setting the SmallCapsFeatures separately for each font shape.

\begin{verbatim}
\fontspec[
            UprightFeatures={Color = 22/zero.noslash/zero.noslash22,
                            SmallCapsFeatures = {Color=115511}},
            ItalicFeatures={Color = 2244FF,
                            SmallCapsFeatures = {Color=112299}},
            BoldFeatures={Color = FF4422,
                          SmallCapsFeatures = {Color=992211}},
            BoldItalicFeatures={Color = 888844,
                                SmallCapsFeatures = {Color=444422}}]{TeX Gyre Termes}
Upright \textsc{Small Caps}\ 
\itshape Italic \textsc{Italic Small Caps}\ 
\upshape\bfseries Bold \textsc{Bold Small Caps}\ 
\upshape\itshape Bold Italic \textsc{Bold Italic Small Caps}
\end{verbatim}

be nested within ItalicFeatures and friends. Every combination of upright, italic, bold and small caps can thus be assigned individual features, as shown in the somewhat ludicrous Example 9.

7.5 Different features for different font sizes

\begin{verbatim}
SizeFeatures = {
    ... ,
    { Size = (size range), \{font features\} },
    { Size = (size range), Font = \{font name\}, \{font features\} },
    ... 
}
\end{verbatim}

The SizeFeature feature is a little more complicated than the previous features discussed. It allows different fonts and different font features to be selected for a given font family as the point size varies.

It takes a comma separated list of braced, comma separated lists of features for each size range. Each sub-list must contain the Size option to declare the size range, and optionally Font to change the font based on size. Other (regular) fontspec features that are added are used on top of the font features that would be used anyway. A demonstration to hopefully clarify these details is shown in Example 10. A less trivial example is shown in the context of optical font sizes in Section 8.6 on page 19.
Example 10: An example of specifying different font features for different sizes of font with \texttt{SizeFeatures}.

```latex
\texttt{\textbackslash fontspec\{} \texttt{SizeFeatures=}{
  \{Size={-8}, Font=\texttt{TeX Gyre Bonum Italic, Color=AA000000}\},
  \{Size={8-14}, Color=00AA0000\},
  \{Size={14-}, Color=0000AA000\} \}[\texttt{TeX Gyre Chorus}]
```

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Input & Font size, s & \\
\hline
Size = X- & s \geq X & \\
Size = -Y & s < Y & \\
Size = X-Y & X \leq s < Y & \\
Size = X & s = X & \\
\hline
\end{tabular}
\caption{Syntax for specifying the size to apply custom font features.}
\label{tab:fontspec}
\end{table}

To be precise, the \texttt{Size} sub-feature accepts arguments in the form shown in Table 1. Braces around the size range are optional. For an exact font size (\texttt{Size=X}) font sizes chosen near that size will ‘snap’. For example, for size definitions at exactly 11pt and 14pt, if a 12pt font is requested actually the 11pt font will be selected. This is a remnant of the past when fonts were designed in metal (at obviously rigid sizes) and later when bitmap fonts were similarly designed for fixed sizes.

If additional features are only required for a single size, the other sizes must still be specified. As in:

\begin{verbatim}
\texttt{SizeFeatures=}{
  \{Size=10,Numbers=Uppercase\},
  \{Size=10-\}}
\end{verbatim}

Otherwise, the font sizes greater than 10 won’t be defined!

\section{Font independent options}

Features introduced in this section may be used with any font.

\subsection{Colour}

\texttt{Color} (or \texttt{Colour}), also shown in Section 7.1 on page 13 and elsewhere, uses font specifications to set the colour of the text. The colour is defined as a triplet of two-digit Hex RGB values, with optionally another value for the transparency (where 00 is completely transparent and FF is opaque.) Transparency is supported by \texttt{Lua\LaTeX} and by \texttt{Xe\LaTeX} with the \texttt{xdv2pdf} driver (Mac OS X only); \texttt{Xe\LaTeX} with the \texttt{xdvipdfmx} driver does not support this feature.

If you load the \texttt{xcolor} package, you may use any named colour instead of writing the colours in hexadecimal.
Example 11: Selecting colour with transparency.

\fontsize{48}{48}\fontspec{TeX Gyre Bonum Bold}
{\addfontfeature{Color=FF000099}W}\kern-1ex
{\addfontfeature{Color=0000FF99}S}\kern-0.8ex
{\addfontfeature{Color=00BB2299}P}\kern-0.8ex
{\addfontfeature{Color=00BB3399}R}

\usepackage{xcolor}
...
\fontspec[Color=red]{Verdana} ...
\definecolor{Foo}{rgb}{0.3,0.4,0.5}
\fontspec[Color=Foo]{Verdana} ...

The color package is not supported; use xcolor instead.

You may specify the transparency with a named colour using the Opacity feature which takes an decimal from zero to one corresponding to transparent to opaque respectively:

\fontspec[Color=red,Opacity=0.7]{Verdana} ...

It is still possible to specify a colour in six-char hexadecimal form while defining opacity in this way, if you like.

8.2 Scale

Scale = \langle number\rangle
Scale = \texttt{MatchLowercase}
Scale = \texttt{MatchUppercase}

In its explicit form, Scale takes a single numeric argument for linearly scaling the font, as demonstrated in Section 5 on page 8. It is now possible to measure the correct dimensions of the fonts loaded and calculate values to scale them automatically.

As well as a numerical argument, the Scale feature also accepts options MatchLowercase and MatchUppercase, which will scale the font being selected to match the current default roman font to either the height of the lower case or uppercase letters, respectively; these features are shown in Example 12.

The amount of scaling used in each instance is reported in the \texttt{.log} file. Since there is some subjectivity about the exact scaling to be used, these values should be used to fine-tune the results.

8.3 Interword space

While the space between words can be varied on an individual basis with the \TeX\ primitive \texttt{\spaceskip} command, it is more convenient to specify this information when the font is first defined.

17
Example 12: Automatically calculated scale values.

The perfect match is hard to find.

Example 13: Scaling the default interword space. An exaggerated value has been chosen to emphasise the effects here.

Some text for our example to take up some space, and to demonstrate the default interword space.

Some text for our example to take up some space, and to demonstrate the default interword space.

The space in between words in a paragraph will be chosen automatically, and generally will not need to be adjusted. For those times when the precise details are important, the WordSpace feature is provided, which takes either a single scaling factor to scale the default value, or a triplet of comma-separated values to scale the nominal value, the stretch, and the shrink of the interword space by, respectively. $(\text{WordSpace}=x)$ is the same as $\text{WordSpace}=(x,x,x)$.

### 8.4 Post-punctuation space

If $\textbackslash \text{frenchspacing}$ is not in effect, \TeX{} will allow extra space after some punctuation in its goal of justifying the lines of text. Generally, this is considered old-fashioned, but occasionally in small amounts the effect can be justified, pardon the pun.

The PunctuationSpace feature takes a scaling factor by which to adjust the nominal value chosen for the font; this is demonstrated in Example 14. Note that $\text{PunctuationSpace}=0$ is not equivalent to $\textbackslash \text{frenchspacing}$, although the difference will only be apparent when a line of text is under-full.

### 8.5 The hyphenation character

The letter used for hyphenation may be chosen with the HyphenChar feature. It takes three types of input, which are chosen according to some simple rules. If the input is the string None, then hyphenation is suppressed for this font. If the input is a single character, then this character is used. Finally, if the input is longer than
Example 14: Scaling the default post-punctuation space.

\nonfrenchspacing
Letters, Words. Sentences. \
\fontspec{TeX Gyre Schola}
Letters, Words. Sentences. \par
\fontspec[PunctuationSpace=2]{TeX Gyre Schola}
Letters, Words. Sentences. \par
\fontspec[PunctuationSpace=0]{TeX Gyre Schola}
Letters, Words. Sentences.

Example 15: Explicitly choosing the hyphenation character.

\def\text{\fbox{\parbox{1.55cm}{EXAMPLE HYPHENATION}}\quad\null\par\bigskip}
\fontspec{Linux Libertine}
\addfontfeature{HyphenChar=None}
\text
\addfontfeature{HyphenChar={+}}
\text

a single character it must be the UTF-8 slot number of the hyphen character you desire.

This package redefines L\LaTeX{}'s \texttt{-} macro such that it adjusts along with the above changes.

8.6 Optical font sizes

Optically scaled fonts thicken out as the font size decreases in order to make the glyph shapes more robust (less prone to losing detail), which improves legibility. Conversely, at large optical sizes the serifs and other small details may be more delicately rendered.

OpenType fonts with optical scaling will exist in several discrete sizes, and these will be selected by X\TeX{} and Lua\TeX{} automatically determined by the current font size as in Example 16, in which we've scaled down some large text in order to be able to compare the difference for equivalent font sizes.

The \texttt{OpticalSize} option may be used to specify a different optical size. With \texttt{OpticalSize} set to zero, no optical size font substitution is performed, as shown in Example 17.

The \texttt{SizeFeatures} feature (Section 7.5 on page 15) can be used to specify exactly which optical sizes will be used for ranges of font size. For example, something like:

```
\fontspec[
  \text{SizeFeatures=}{
    \{Size=-10, \quad OpticalSize=8\} },
```
Part II

OpenType

9 Introduction

OpenType fonts (and other ‘smart’ font technologies such as AAT and Graphite) can change the appearance of text in many different ways. These changes are referred to as features. When the user applies a feature — for example, small capitals — to a run of text, the code inside the font makes appropriate adjustments and small capitals appear in place of lowercase letters. However, the use of such features does not affect the underlying text. In our small caps example, the lowercase letters are still stored in the document; only the appearance has been changed by the OpenType feature. This makes it possible to search and copy text without difficulty. If the user selected a different font that does not support small caps, the ‘plain’ lowercase letters would appear instead.

Some OpenType features are required to support particular scripts, and these features are often applied automatically. The scripts used in India, for example, often require that characters be reshaped and reordered after they are typed by the user, in order to display them in the traditional ways that readers expect. Other features can be applied to support a particular language. The Junicode font for
medievalists uses by default the Old English shape of the letter thorn, while in modern Icelandic thorn has a more rounded shape. If a user tags some text as being in Icelandic, Unicode will automatically change to the Icelandic shape through an OpenType feature that localizes the shapes of letters.

A very large group of OpenType features is designed to support high quality typography in Latin, Greek, Cyrillic and other standard scripts. Examples of some font features have already been shown in previous sections; the complete set of OpenType font features supported by fontspec is described below in Section 10.

The OpenType specification provides four-letter codes (e.g., `smcp` for small capitals) for each feature. The four-letter codes are given below along with the fontspec names for various features, for the benefit of people who are already familiar with OpenType. You can ignore the codes if they don’t mean anything to you.

9.1 How to select font features

Font features are selected by a series of `(feature)=(option)` selections. Features are (usually) grouped logically; for example, all font features relating to ligatures are accessed by writing `Ligatures={...}` with the appropriate argument(s), which could be `TeX`, `Rare`, etc., as shown below in Section 10.1.

Multiple options may be given to any feature that accepts non-numerical input, although doing so will not always work. Some options will override others in generally obvious ways; `Numbers={OldStyle,Lining}` doesn’t make much sense because the two options are mutually exclusive, and `XeLaTeX` will simply use the last option that is specified (in this case using `Lining` over `OldStyle`).

If a feature or an option is requested that the font does not have, a warning is given in the console output. As mentioned in Section 3.3 on page 5 these warnings can be suppressed by selecting the `[quiet]` package option.

10 Complete listing of OpenType font features

10.1 Ligatures

Ligatures refer to the replacement of two separate characters with a specially drawn glyph for functional or aesthetic reasons. The list of options, of which multiple may be selected at one time, is shown in Table 2. A demonstration with the Linux Libertine fonts\(^2\) is shown in Example 18.

Note the additional features accessed with `Ligatures=TeX`. These are not actually real OpenType features, but additions provided by `luaotfload` (i.e., LuaLaTeX only) to emulate `TeX`’s behaviour for `ascii` input of curly quotes and punctuation. In `XeLaTeX` this is achieved with the `Mapping` feature (see Section 12.1 on page 39) but for consistency `Ligatures=TeX` will perform the same function as `Mapping=\text{-text}`.

\(^2\)http://www.linuxlibertine.org/
Table 2: Options for the OpenType font feature ‘Ligatures’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ligatures</td>
<td>Required</td>
<td>rlig</td>
</tr>
<tr>
<td></td>
<td>NoRequired</td>
<td>rlig (deactivate)</td>
</tr>
<tr>
<td>Common</td>
<td></td>
<td>liga</td>
</tr>
<tr>
<td>NoCommon</td>
<td></td>
<td>liga (deactivate)</td>
</tr>
<tr>
<td>Contextual</td>
<td></td>
<td>clig</td>
</tr>
<tr>
<td>NoContextual</td>
<td></td>
<td>clig (deactivate)</td>
</tr>
<tr>
<td>Rare/Discretionary</td>
<td></td>
<td>dlg</td>
</tr>
<tr>
<td>Historic</td>
<td></td>
<td>hlig</td>
</tr>
<tr>
<td>TeX</td>
<td></td>
<td>tlig/trep</td>
</tr>
</tbody>
</table>

* This feature is activated by default.

Example 18: An example of the Ligatures feature.

\def\test#1#2{%
 #2 $\to$ \{\addfontfeature{#1} #2\}\}
\fontspec{Linux Libertine}
\test{Ligatures=Historic}{strict}
\test{Ligatures=Rare}{wurtzite}
\test{Ligatures=NoCommon}{firefly}

\begin{verbatim}
strict \rightarrow \textit{strict}
wurtzite \rightarrow \textit{wurtzite}
firefly \rightarrow \textit{firefly}
\end{verbatim}
Table 3: Options for the OpenType font feature ‘Letters’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letters =</td>
<td>Uppercase</td>
<td>case</td>
</tr>
<tr>
<td></td>
<td>SmallCaps</td>
<td>smcp</td>
</tr>
<tr>
<td></td>
<td>PetiteCaps</td>
<td>pcap</td>
</tr>
<tr>
<td></td>
<td>UppercaseSmallCaps</td>
<td>c2sc</td>
</tr>
<tr>
<td></td>
<td>UppercasePetiteCaps</td>
<td>c2pc</td>
</tr>
<tr>
<td></td>
<td>Unicase</td>
<td>unic</td>
</tr>
</tbody>
</table>

Example 19: Small caps from lowercase or uppercase letters.

\fontspec[Letters=SmallCaps]{TeX Gyre Adventor}
\fontspec[Letters=UppercaseSmallCaps]{TeX Gyre Adventor}

10.2 Letters

The Letters feature specifies how the letters in the current font will look. OpenType fonts may contain the following options: Uppercase, SmallCaps, PetiteCaps, UppercaseSmallCaps, UppercasePetiteCaps, and Unicase.

Petite caps are smaller than small caps. SmallCaps and PetiteCaps turn lowercase letters into the smaller caps letters, whereas the Uppercase... options turn the capital letters into the smaller caps (good, e.g., for applying to already uppercase acronyms like ‘NASA’). This difference is shown in Example 19. ‘Unicase’ is a weird hybrid of upper and lower case letters.

Note that the Uppercase option will (probably) not actually map letters to uppercase. It is designed to select various uppercase forms for glyphs such as accents and dashes, such as shown in Example 20; note the raised position of the hyphen to better match the surrounding letters.

The Kerning feature also contains an Uppercase option, which adds a small amount of spacing in between letters (see Section 10.12 on page 30).

3If you want automatic uppercase letters, look to $\LaTeX$’s \MakeUppercase command.

Example 20: An example of the Uppercase option of the Letters feature.

\fontspec{Linux Libertine}
\addfontfeature{Letters=Uppercase}

\textsc{uppercase} example
\textsc{uppercase} example

23
10.3 Numbers

The Numbers feature defines how numbers will look in the selected font, accepting options shown in Table 4.

The synonyms Uppercase and Lowercase are equivalent to Lining and OldStyle, respectively. The differences have been shown previously in Section 7.2 on page 13. The Monospaced option is useful for tabular material when digits need to be vertically aligned.

The SlashedZero option replaces the default zero with a slashed version to prevent confusion with an uppercase ‘O’, shown in Example 21.

The Arabic option (with tag anum) maps regular numerals to their Arabic script or Persian equivalents based on the current Language setting (see Section 10.18 on page 35), shown in Example 22 using the Persian Modern font, which is included in \TeX Live and MiK\TeX. This option is based on a \Lua\TeX feature of the \luaotfload package, not an OpenType feature. (Thus, this feature is unavailable in \X\TeX.)

---

Table 4: Options for the OpenType font feature ‘Numbers’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>Uppercase/Lining</td>
<td>lnum</td>
</tr>
<tr>
<td></td>
<td>Lowercase/OldStyle</td>
<td>onum</td>
</tr>
<tr>
<td></td>
<td>Proportional</td>
<td>pnum</td>
</tr>
<tr>
<td></td>
<td>Monospaced</td>
<td>tnum</td>
</tr>
<tr>
<td></td>
<td>SlashedZero</td>
<td>zero</td>
</tr>
<tr>
<td></td>
<td>Arabic</td>
<td>anum</td>
</tr>
</tbody>
</table>

---

Example 21: The effect of the SlashedZero option.

\fontspec[^Numbers=Lining]{TeX Gyre Bonum} 0123456789
\fontspec[^Numbers=SlashedZero]{TeX Gyre Bonum} 0123456789

---

Example 22: An example of number remapping to Arabic or Persian. (\Lua\TeX only.)

\fontspec[^Script=Arabic,^Numbers=Arabic] (persion-modern-regular.ttf)  \
\{\addfontfeature[^Language=Arabic] 0123456789\}
\{\addfontfeature[^Language=Parsi] 0123456789\}

---

24
Table 5: Options for the OpenType font feature ‘Contextuals’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contextuals</td>
<td>Swash</td>
<td>cswh</td>
</tr>
<tr>
<td></td>
<td>Alternate</td>
<td>calt</td>
</tr>
<tr>
<td></td>
<td>WordInitial</td>
<td>init</td>
</tr>
<tr>
<td></td>
<td>WordFinal</td>
<td>fina</td>
</tr>
<tr>
<td></td>
<td>LineFinal</td>
<td>falt</td>
</tr>
<tr>
<td></td>
<td>Inner</td>
<td>medi</td>
</tr>
</tbody>
</table>

Example 23: An example of the Swashes option of the Contextuals feature.

\texttt{\fontspec{Warnock Pro} \itshape}
\texttt{Without Contextual Swashes}
\texttt{\fontspec[Contextuals=Swash]{Warnock Pro}}
\texttt{With Contextual Swashes; cf. W C S}

10.4 Contextuals

This feature refers to substitutions of glyphs that vary ‘contextually’ by their relative position in a word or string of characters; features such as contextual swashes are accessed via the options shown in Table 5. See Example 23 for an, er, example.

Historic forms are accessed in OpenType fonts via the feature \texttt{Style=Historic}; this is generally \textit{not} contextual in OpenType, which is why it is not included here.

10.5 Vertical Position

The \texttt{VerticalPosition} feature is used to access things like subscript (\texttt{Inferior}) and superscript (\texttt{Superior}) numbers and letters (and a small amount of punctuation, sometimes). The \texttt{Ordinal} option will only raise characters that are used in some languages directly after a number. The \texttt{ScientificInferior} feature will move glyphs further below the baseline than the \texttt{Inferior} feature. These are shown in Example 24.

\texttt{Numerator} and \texttt{Denominator} should only be used for creating arbitrary fractions (see next section).

Table 6: Options for the OpenType font feature ‘VerticalPosition’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>VerticalPosition</td>
<td>Superior</td>
<td>sups</td>
</tr>
<tr>
<td></td>
<td>Inferior</td>
<td>subs</td>
</tr>
<tr>
<td></td>
<td>Numerator</td>
<td>numr</td>
</tr>
<tr>
<td></td>
<td>Denominator</td>
<td>dnom</td>
</tr>
<tr>
<td></td>
<td>ScientificInferior</td>
<td>sinf</td>
</tr>
<tr>
<td></td>
<td>Ordinal</td>
<td>ordn</td>
</tr>
</tbody>
</table>
Example 24: The VerticalPosition feature. Note that the Ordinal option can be quite unreliable, as the results here demonstrate.

<table>
<thead>
<tr>
<th>Sup: abdehilmnorst (-$12,345.67)</th>
<th>\</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsuperscript{Sup: abdehilmnorst (-$12,345.67)}</td>
<td>\</td>
</tr>
<tr>
<td>\textsuperscript{Numerator: 12345}</td>
<td>\</td>
</tr>
<tr>
<td>\textsuperscript{Denominator: 12345}</td>
<td>\</td>
</tr>
<tr>
<td>\textsuperscript{Scientific Inferior: 12345}</td>
<td>\</td>
</tr>
<tr>
<td>‘Ordinals’: 1\textsuperscript{st} 2\textsuperscript{nd} 3\textsuperscript{rd} 4\textsuperscript{th} /zero.noslash</td>
<td>\</td>
</tr>
</tbody>
</table>

Table 7: Options for the OpenType font feature ‘Fractions’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fractions = On</td>
<td>frac</td>
<td></td>
</tr>
<tr>
<td>Alternate</td>
<td>afrc</td>
<td></td>
</tr>
</tbody>
</table>

The realscripts package (which is also loaded by \textxtra for X\TeX) redefines the \textsubscript and \textsuperscript commands to use the above font features automatically, including for use in footnote labels. If this is the only feature of \textxtra you wish to use, consider loading realscripts on its own instead.

10.6 Fractions

For OpenType fonts use a regular text slash to create fractions, but the Fraction feature must be explicitly activated. Some (Asian fonts predominantly) also provide for the Alternate feature. These are both shown in Example 25.

10.7 Stylistic Set variations

This feature selects a ‘Stylistic Set’ variation, which usually corresponds to an alternate glyph style for a range of characters (usually an alphabet or subset thereof).

Example 25: The Fractions feature.

\begin{verbatim}
\fontspec[Hiragino Maru Gothic Pro W4]
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \quad \\addfontfeature{Fractions=On}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \quad \\addfontfeature{Fractions=Alternate}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \quad \\addfontfeature{Fractions=On}
1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 \quad \\addfontfeature{Fractions=Alternate}
\end{verbatim}
Example 26: Insular letterforms, as used in medieval Northern Europe, for the Junicode font accessed with the StylisticSet feature.

\begin{align*}
\text{Insular forms.} & \quad \text{\fontspec{Junicode}} \\
\text{Inꞅulaꞃ ꦺoꞃmꞅ.} & \quad \text{\addfontfeature{StylisticSet=2}} \\
\end{align*}

Example 27: Enlarged minuscules (capital letters remain unchanged) for the Junicode font, accessed with the StylisticSet feature.

\begin{align*}
\text{ENLARGED Minuscules.} & \quad \text{\fontspec{Junicode}} \\
\text{ENLARGED Minuscules.} & \quad \text{\addfontfeature{StylisticSet=6}}
\end{align*}

This feature is specified numerically. These correspond to OpenType features ss01, ss02, etc.

Two demonstrations from the Junicode font\textsuperscript{4} are shown in Example 26 and Example 27; thanks to Adam Buchbinder for the suggestion.

Multiple stylistic sets may be selected simultaneously by writing, e.g.,\texttt{StylisticSet=\{1,2,3\}}.

The StylisticSet feature is a synonym of the Variant feature for AAT fonts.

See Section 15 on page 46 for a way to assign names to stylistic sets, which should be done on a per-font basis.

10.8 Character Variants

Similar to the ‘Stylistic Sets’ above, ‘Character Variations’ are selected numerically to adjust the output of (usually) a single character for the particular font. These correspond to the OpenType features cv01 to cv99.

For each character that can be varied, it is possible to select among possible options for that particular glyph. For example, in Example 28 a variety of glyphs for the character ‘v’ are selected, in which 5 corresponds to the character ‘v’ for this font feature, and the trailing \texttt{:⟨n⟩} corresponds to which variety to choose. Georg Duffner’s open source Garamond revival font\textsuperscript{5} is used in this example. Character variants are specifically designed not to conflict with each other, so you can enable them individually per character as shown in Example 29. (Unlike stylistic alternates, say.)

Note that the indexing starts from zero, which is compatible with \LaTeX{} but \texttt{luaotfload}, which starts from one.

10.9 Alternates

The Alternate feature (for the raw OpenType feature \texttt{salt}) is used to access alternate font glyphs when variations exist in the font, such as in Example 30. It uses a

\texttt{http://junicode.sf.net}  \\
\texttt{http://www.georgduffner.at/ebgaramond/}
Example 28: The CharacterVariant feature showing off Georg Duffner’s open source Garamond revival font.

\texttt{very}

\texttt{very}

\texttt{very}

\texttt{very}

\texttt{very}

\texttt{very}

\texttt{\textbackslash fonts[EB Garamond Italic] very \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant=5][EB Garamond Italic] very \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant=5:0][EB Garamond Italic] very \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant=5:1][EB Garamond Italic] very \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant=5:2][EB Garamond Italic] very \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant=5:3][EB Garamond Italic] very}\n

Example 29: The CharacterVariant feature selecting multiple variants simultaneously.

\texttt{\& violet}

\texttt{\& violet}

\texttt{\& violet}

\texttt{\& violet}

\texttt{\textbackslash fonts[EB Garamond Italic] \& violet \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant={4}][EB Garamond Italic] \& violet \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant={5:2}][EB Garamond Italic] \& violet \textbackslash}\n
\texttt{\textbackslash fonts[CharacterVariant={4,5:2}][EB Garamond Italic] \& violet}
Example 30: The *Alternate* feature.

| A & h |
| A • h |

\texttt{\texttt{\fontspec{Linux Libertine}} \texttt{\textsc{a} \& h \textbackslash}}
\texttt{\texttt{\texttt{\addfontfeature{Alternate=0}} \texttt{\texttt{\textsc{a} \& h}}}}

---

**Table 8:** Options for the OpenType font feature ‘Style’.

<table>
<thead>
<tr>
<th>Feature Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Style</em> = Alternate</td>
<td>salt</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>ital</td>
</tr>
<tr>
<td><em>Ruby</em></td>
<td>ruby</td>
</tr>
<tr>
<td><em>Swash</em></td>
<td>swsh</td>
</tr>
<tr>
<td><em>Historic</em></td>
<td>hist</td>
</tr>
<tr>
<td><em>TitlingCaps</em></td>
<td>titl</td>
</tr>
<tr>
<td><em>HorizontalKana</em></td>
<td>hkna</td>
</tr>
<tr>
<td><em>VerticalKana</em></td>
<td>vkna</td>
</tr>
</tbody>
</table>

Numerical selection, starting from zero, that will be different for each font. Note that the *Style=Alternate* option is equivalent to *Alternate=0* to access the default case.

Note that the indexing starts from zero, which is compatible with plain \texttt{Xe\TeX} but *incompatible* with \texttt{luaotfload}, which starts from one.

See Section 15 on page 46 for a way to assign names to alternates, which must be done on a per-font basis.

### 10.10 Style

‘Ruby’ refers to a small optical size, used in Japanese typography for annotations. For fonts with multiple \texttt{salt} OpenType features, use the \texttt{\fontspec{Alternate}} feature instead.

Example 31 and Example 32 both contain glyph substitutions with similar characteristics. Note the occasional inconsistency with which font features are labelled; a long-tailed ‘Q’ could turn up anywhere!

In other features, larger breadths of changes can be seen, covering the style of an entire alphabet. See Example 33 and Example 34; in the latter, the *Italic* option

---

Example 31: Example of the *Alternate* option of the *Style* feature.

\texttt{\texttt{\fontspec{Warnock Pro}} \texttt{\texttt{\texttt{K Q R k v w y \textbackslash}}\texttt{\texttt{\addfontfeature{Style=Alternate}}} \texttt{\texttt{\texttt{K Q R k v w y}}}}}

29
Example 32: Example of the Historic option of the Style feature.

\fontspec{Adobe Jenson Pro}
\addfontfeature{Style=Historic}
M Q Z

Example 33: Example of the TitlingCaps option of the Style feature.

\fontspec{Adobe Garamond Pro}
\addfontfeature{Style=TitlingCaps}
TITLING CAPS

affects the Latin text and the Ruby option the Japanese.
Note the difference here between the default and the horizontal style kana in Example 35: the horizontal style is slightly wider.

10.11 Diacritics
Specifies how combining diacritics should be placed. These will usually be controlled automatically according to the Script setting.

10.12 Kerning
Specifies how inter-glyph spacing should behave. Well-made fonts include information for how differing amounts of space should be inserted between separate character pairs. This kerning space is inserted automatically but in rare circumstances you may wish to turn it off.

As briefly mentioned previously at the end of Section 10.2 on page 23, the Uppercase option will add a small amount of tracking between uppercase letters, seen in Example 36, which uses the Romande fonts⁶ (thanks to Clea F. Rees for the suggestion). The Uppercase option acts separately to the regular kerning controlled by the On/Off options.

Example 34: Example of the Italic and Ruby options of the Style feature.

\fontspec{Hiragino Mincho Pro}
\addfontfeature{Style={Italic, Ruby}}
Latin ようこそ ワカヨタレソ

\fontspec{Hiragino Mincho Pro}
\addfontfeature{Style={Italic, Ruby}}
Latin ようこそ ワカヨタレソ

⁶http://arkandis.tuxfamily.org/adffonts.html
Example 35: Example of the HorizontalKana and VerticalKana options of the Style feature.

Table 9: Options for the OpenType font feature ‘Diacritics’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diacritics</td>
<td>MarkToBase</td>
<td>mark</td>
</tr>
<tr>
<td></td>
<td>NoMarkToBase</td>
<td>mark (deactivate)</td>
</tr>
<tr>
<td>MarkToMark</td>
<td>mkmk</td>
<td></td>
</tr>
<tr>
<td>NoMarkToMark</td>
<td>mkmk (deactivate)</td>
<td></td>
</tr>
<tr>
<td>AboveBase</td>
<td>abvm</td>
<td></td>
</tr>
<tr>
<td>NoAboveBase</td>
<td>abvm (deactivate)</td>
<td></td>
</tr>
<tr>
<td>BelowBase</td>
<td>blwm</td>
<td></td>
</tr>
<tr>
<td>NoBelowBase</td>
<td>blwm (deactivate)</td>
<td></td>
</tr>
</tbody>
</table>

* This feature is activated by default.

Table 10: Options for the OpenType font feature ‘Kerning’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kerning</td>
<td>Uppercase</td>
<td>cpsp</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>kern</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>kern (deactivate)</td>
</tr>
</tbody>
</table>

* This feature is activated by default.

Example 36: Adding extra kerning for uppercase letters. (The difference is usually very small.)

```latex
\textbf{UPPERCASE EXAMPLE}

\textbf{UPPERCASE EXAMPLE}
```

31
10.13 Font transformations

In rare situations users may want to mechanically distort the shapes of the glyphs in the current font such as shown in Example 37. Please don’t overuse these features; they are not a good alternative to having the real shapes.

If values are omitted, their defaults are as shown above.

If you want the bold shape to be faked automatically, or the italic shape to be slanted automatically, use the AutoFakeBold and AutoFakeSlant features. For example, the following two invocations are equivalent:

\fontspec[AutoFakeBold=1.5]{Charis SIL}
\fontspec[BoldFeatures={FakeBold=1.5}]{Charis SIL}

If both of the AutoFake... features are used, then the bold italic font will also be faked.

The FakeBold and AutoFakeBold features are only available with the X\TeX engine and will be ignored in Lua\TeX.

10.14 Annotation

Some fonts are equipped with an extensive range of numbers and numerals in different forms. These are accessed with the Annotation feature (OpenType feature nalt), selected numerically as shown in Example 38.

Note that the indexing starts from zero, which is compatible with X\TeX but incompatible with luaotfload, which starts from one.

10.15 CJK shape

There have been many standards for how CJK ideographic glyphs are ‘supposed’ to look. Some fonts will contain many alternate glyphs available in order to be able to display these glyphs correctly in whichever form is appropriate. Both AAT and OpenType fonts support the following CJKShape options: Traditional, Simplified, JIS1978, JIS1983, JIS1990, and Expert. OpenType also supports the NLC option.

10.16 Character width

Many Asian fonts are equipped with variously spaced characters for shoe-horning into their generally monospaced text. These are accessed through the
Example 38: Annotation forms for OpenType fonts.

```
1 2 3 4 5 6 7 8 9
(1 2 3 (4 5 6 7 8 9)
(1 2 3 4 5 6 7 8 9)
(1 2 3 4 5 6 7 8 9)
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

\text{Example 39: Different standards for CJK ideograph presentation.}

```
\text{唗嘠驅 妍并訥}
\text{唗嘠驅 妍并訥}
\text{啞嘠驅 妍并訥}
```

Table 11: Options for the OpenType font feature ‘CJKShape’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJKShape = Traditional</td>
<td>trad</td>
<td>1</td>
</tr>
<tr>
<td>Simplified</td>
<td>smp1</td>
<td>2</td>
</tr>
<tr>
<td>JIS1978</td>
<td>jp78</td>
<td>3</td>
</tr>
<tr>
<td>JIS1983</td>
<td>jp83</td>
<td>4</td>
</tr>
<tr>
<td>JIS1990</td>
<td>jp90</td>
<td>5</td>
</tr>
<tr>
<td>Expert</td>
<td>expt</td>
<td>6</td>
</tr>
<tr>
<td>NLC</td>
<td>nlc</td>
<td>7</td>
</tr>
</tbody>
</table>

```
\text{Example 39: Different standards for CJK ideograph presentation.}

```
\text{唗嘠驅 妍并訥}
\text{唗嘠驅 妍并訥}
\text{啞嘠驅 妍并訥}
```

\text{33}
Table 12: Options for the OpenType font feature ‘CharacterWidth’.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharacterWidth</td>
<td>Proportional</td>
<td>pwid</td>
</tr>
<tr>
<td></td>
<td>Full</td>
<td>fwid</td>
</tr>
<tr>
<td></td>
<td>Half</td>
<td>hwid</td>
</tr>
<tr>
<td></td>
<td>Third</td>
<td>twid</td>
</tr>
<tr>
<td></td>
<td>Quarter</td>
<td>qwid</td>
</tr>
<tr>
<td></td>
<td>AlternateProportional</td>
<td>palt</td>
</tr>
<tr>
<td></td>
<td>AlternateHalf</td>
<td>halt</td>
</tr>
</tbody>
</table>

Example 40: Proportional or fixed width forms.

\def\test{\makebox[2cm][l]{texta}\makebox[2.5cm][l]{\textb}\makebox[2.5cm][l]{abcdef}}\fontspec{Hiragino Mincho Pro}{\addfontfeature{CharacterWidth=Proportional}\test}\addfontfeature{CharacterWidth=Full}\test\addfontfeature{CharacterWidth=Half}\test

CharacterWidth feature.

Japanese alphabetic glyphs (in Hiragana or Katakana) may be typeset proportionally, to better fit horizontal measures, or monospaced, to fit into the rigid grid imposed by ideographic typesetting. In this latter case, there are also half-width forms for squeezing more kana glyphs (which are less complex than the kanji they are amongst) into a given block of space. The same features are given to roman letters in Japanese fonts, for typesetting foreign words in the same style as the surrounding text.

The same situation occurs with numbers, which are provided in increasingly illegible compressed forms seen in Example 41.

Example 41: Numbers can be compressed significantly.

\fontspec[Renderer=AAT]{Hiragino Mincho Pro}\addfontfeature{CharacterWidth=Full}\addfontfeature{CharacterWidth=Half}\addfontfeature{CharacterWidth=Third}\addfontfeature{CharacterWidth=Quarter}

34
10.17 Vertical typesetting

TODO!

10.18 OpenType scripts and languages

Fonts that include glyphs for various scripts and languages may contain different font features for the different character sets and languages they support, and different font features may behave differently depending on the script or language chosen. When multilingual fonts are used, it is important to select which language they are being used for, and more importantly what script is being used.

The ‘script’ refers to the alphabet in use; for example, both English and French use the Latin script. Similarly, the Arabic script can be used to write in both the Arabic and Persian languages.

The Script and Language features are used to designate this information. The possible options are tabulated in Table 13 on the following page and Table 14 on page 37, respectively. When a script or language is requested that is not supported by the current font, a warning is printed in the console output.

Because these font features can change which features are able to be selected for the font, they are automatically selected by fontspec before all others and, if Xe\LaTeX is being used, will specifically select the ICU renderer for this font, as described in Section 12.3 on page 40.

10.18.1 Script and Language examples

In the examples shown in Example 42, the Code2000 font is used to typeset various input texts with and without the OpenType Script applied for various alphabets. The text is only rendered correctly in the second case; many examples of incorrect diacritic spacing as well as a lack of contextual ligatures and rearrangement can be seen. Thanks to Jonathan Kew, Yves Codet and Gildas Hamel for their contributions towards these examples.

10.18.2 Defining new scripts and languages

While the scripts and languages listed in Table 13 and Table 14 are intended to be comprehensive, there may be some missing; alternatively, you might wish to use different names to access scripts/languages that are already listed. Adding scripts and languages can be performed with the \newfontscript and \newfontlanguage commands. For example,

\newfontscript{Arabic}{arab}
\newfontlanguage{Zulu}{ZUL}

The first argument is the fontspec name, the second the OpenType tag. The advantage to using these commands rather than \newfontfeature (see Section 15 on page 46) is the error-checking that is performed when the script or language is requested.

\footnote{http://www.code2000.net/}
Example 42: An example of various Scripts and Languages.

Table 13: Defined Scripts for OpenType fonts. Aliased names are shown in adjacent positions marked with red pilcrows (¶).

<table>
<thead>
<tr>
<th>Arabic</th>
<th>Ethiopic</th>
<th>Limbu</th>
<th>Sumero-Akkadian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Armenian</td>
<td>Georgian</td>
<td>Linear B</td>
<td>Cuneiform</td>
</tr>
<tr>
<td>Balinese</td>
<td>Glagoltic</td>
<td>Malayalam</td>
<td>Syloti Nogri</td>
</tr>
<tr>
<td>Bengali</td>
<td>Gothic</td>
<td>Math</td>
<td>Syriac</td>
</tr>
<tr>
<td>Bopomofo</td>
<td>Greek</td>
<td>Math</td>
<td>Tagalog</td>
</tr>
<tr>
<td>Braille</td>
<td>Gujarati</td>
<td>Mongolian</td>
<td>Tagbanwa</td>
</tr>
<tr>
<td>Buginese</td>
<td>Gurmukhi</td>
<td>Musical Symbols</td>
<td>Tai Le</td>
</tr>
<tr>
<td>Buhid</td>
<td>Hangul Jamo</td>
<td>Myanmar</td>
<td>Tamil</td>
</tr>
<tr>
<td>Byzantine Music</td>
<td>Hangul</td>
<td>N’ko</td>
<td>Telugu</td>
</tr>
<tr>
<td>Canadian Syllabics</td>
<td>Hanunoo</td>
<td>Ogham</td>
<td>Thana</td>
</tr>
<tr>
<td>Cherokee</td>
<td>Hebrew</td>
<td>Old Italian</td>
<td>Thai</td>
</tr>
<tr>
<td>CJK</td>
<td>Hira&amp;Katakana</td>
<td>Old Persian Cuneiform</td>
<td>Tibetan</td>
</tr>
<tr>
<td>CJK I</td>
<td>Kana</td>
<td>Onya</td>
<td>Tifinagh</td>
</tr>
<tr>
<td>Coptic</td>
<td>Javanes</td>
<td>Osmanya</td>
<td>Ugarit Cuneiform</td>
</tr>
<tr>
<td>Cyriolic</td>
<td>Kannada</td>
<td>Phag-s-pa</td>
<td>Yi</td>
</tr>
<tr>
<td>Default</td>
<td>Kharosthi</td>
<td>Phoenician</td>
<td></td>
</tr>
<tr>
<td>Deseret</td>
<td>Khmer</td>
<td>Shavian</td>
<td></td>
</tr>
<tr>
<td>Devanagari</td>
<td>Lao</td>
<td>Sinhala</td>
<td></td>
</tr>
</tbody>
</table>

36
Table 14: Defined Languages for OpenType fonts. Aliased names are shown in adjacent positions marked with red pilcrows (¶).

<table>
<thead>
<tr>
<th>Language</th>
<th>Default</th>
<th>Igbo</th>
<th>Koryak</th>
<th>Norway House Cree</th>
<th>Serer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abaza</td>
<td>Default</td>
<td>Igbo</td>
<td>Koryak</td>
<td>Norway House Cree</td>
<td>Serer</td>
</tr>
<tr>
<td>Abkhazian</td>
<td>Dogri</td>
<td>Ijo</td>
<td>Ladin</td>
<td>Nisi</td>
<td>South Slavey</td>
</tr>
<tr>
<td>Adyghe</td>
<td>Divehi</td>
<td>Ilokano</td>
<td>Lahuli</td>
<td>Niuean</td>
<td>Southern Sami</td>
</tr>
<tr>
<td>Afrikaans</td>
<td>Djerman</td>
<td>Indonisian</td>
<td>Lak</td>
<td>Nkole</td>
<td>Suri</td>
</tr>
<tr>
<td>Alar</td>
<td>Dangme</td>
<td>Ingush</td>
<td>Lambani</td>
<td>N'ko</td>
<td>Svan</td>
</tr>
<tr>
<td>Agaw</td>
<td>Dinka</td>
<td>Inuktitut</td>
<td>Lao</td>
<td>Dutch</td>
<td>Swedish</td>
</tr>
<tr>
<td>Altai</td>
<td>Dungan</td>
<td>Irish</td>
<td>Latin</td>
<td>Nogai</td>
<td>Swadaya Aramaic</td>
</tr>
<tr>
<td>Amharic</td>
<td>Dzongkha</td>
<td>Irish Traditional</td>
<td>Laz</td>
<td>Norwegian</td>
<td>Swahili</td>
</tr>
<tr>
<td>Arabian</td>
<td>Ebira</td>
<td>Icelandic</td>
<td>L-Cree</td>
<td>Northern Sami</td>
<td>Swazi</td>
</tr>
<tr>
<td>Aari</td>
<td>Eastern Cree</td>
<td>Inari Sami</td>
<td>Ladakhi</td>
<td>Northern Tai</td>
<td>Sutu</td>
</tr>
<tr>
<td>Arakanese</td>
<td>Edo</td>
<td>Italian</td>
<td>Legzi</td>
<td>Esperanto</td>
<td>Syriac</td>
</tr>
<tr>
<td>Assamese</td>
<td>Efik</td>
<td>Hebrew</td>
<td>Lingala</td>
<td>Nynorsk</td>
<td>Tabasarani</td>
</tr>
<tr>
<td>Athapaskan</td>
<td>Greek</td>
<td>Javanese</td>
<td>Low Mari</td>
<td>Oj-Cree</td>
<td>Tajiki</td>
</tr>
<tr>
<td>Avar</td>
<td>English</td>
<td>Yiddish</td>
<td>Limbu</td>
<td>Ojibway</td>
<td>Tamil</td>
</tr>
<tr>
<td>Awadhi</td>
<td>Erzya</td>
<td>Japanese</td>
<td>Lomwe</td>
<td>Oriya</td>
<td>Tatar</td>
</tr>
<tr>
<td>Aymara</td>
<td>Spanish</td>
<td>Judezmo</td>
<td>Lower Sorbian</td>
<td>Oromo</td>
<td>TH-Cree</td>
</tr>
<tr>
<td>Azeri</td>
<td>Estonian</td>
<td>Jula</td>
<td>Lule Sami</td>
<td>Ossetian</td>
<td>Telugu</td>
</tr>
<tr>
<td>Badaga</td>
<td>Basque</td>
<td>Kabardian</td>
<td>Lithuanian</td>
<td>Pali</td>
<td>Tigre</td>
</tr>
<tr>
<td>Baghelkhandi</td>
<td>Evenki</td>
<td>Kachchi</td>
<td>Luba</td>
<td>Pula</td>
<td>Tigrinya</td>
</tr>
<tr>
<td>Balkar</td>
<td>Even</td>
<td>Kalenjin</td>
<td>Luganda</td>
<td>Punjabi</td>
<td>Thai</td>
</tr>
<tr>
<td>Baule</td>
<td>Ewe</td>
<td>Kannada</td>
<td>Luhy</td>
<td>Palpa</td>
<td>Thai</td>
</tr>
<tr>
<td>Berber</td>
<td>English</td>
<td>Karachay</td>
<td>Lu</td>
<td>Pashto</td>
<td>Tahitian</td>
</tr>
<tr>
<td>Bench</td>
<td>Farsi</td>
<td>Georgian</td>
<td>Latvian</td>
<td>Polytonic Greek</td>
<td>Tibetan</td>
</tr>
<tr>
<td>Bible Cree</td>
<td>Parsi</td>
<td>Kazakh</td>
<td>Majang</td>
<td>Pilipino</td>
<td>Turkmen</td>
</tr>
<tr>
<td>Belarusian</td>
<td>Persian</td>
<td>Kebeina</td>
<td>Makua</td>
<td>Palaung</td>
<td>Temne</td>
</tr>
<tr>
<td>Bemba</td>
<td>Finnish</td>
<td>Khutsuri Georgian</td>
<td>Malayalam</td>
<td>Tungarian</td>
<td>Tonga</td>
</tr>
<tr>
<td>Bengali</td>
<td>Fijian</td>
<td>Khakass</td>
<td>MANSI</td>
<td>Portuguese</td>
<td>Tonga</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Flemish</td>
<td>Khanty-Kazim</td>
<td>Khmer</td>
<td>Marathi</td>
<td>Chin</td>
</tr>
<tr>
<td>Bhili</td>
<td>Forest Nenets</td>
<td>Khanty-Shurishkar</td>
<td>Marwari</td>
<td>Rajasthani</td>
<td>Turkish</td>
</tr>
<tr>
<td>Bhojpuri</td>
<td>Fon</td>
<td>Khanty-Vakhki</td>
<td>Mbundu</td>
<td>R-Cree</td>
<td>Tswana</td>
</tr>
<tr>
<td>Bikol</td>
<td>Faraese</td>
<td>Khmer</td>
<td>Me'en</td>
<td>Romanian</td>
<td>Twi</td>
</tr>
<tr>
<td>Bilen</td>
<td>French</td>
<td>Khowar</td>
<td>Me'nd</td>
<td>Romany</td>
<td>Udmurt</td>
</tr>
<tr>
<td>Blackfoot</td>
<td>Frisian</td>
<td>Kikuyu</td>
<td>Mende</td>
<td>Russian</td>
<td>Tungarian</td>
</tr>
<tr>
<td>Balochi</td>
<td>Friulian</td>
<td>Kirghiz</td>
<td>Me'n</td>
<td>Russian</td>
<td>Uyghur</td>
</tr>
<tr>
<td>Balante</td>
<td>Futa</td>
<td>Kisi</td>
<td>Me'en</td>
<td>Sadri</td>
<td>Uzbek</td>
</tr>
<tr>
<td>Balti</td>
<td>Fulani</td>
<td>Kokni</td>
<td>Mizo</td>
<td>Romany</td>
<td>Udmurt</td>
</tr>
<tr>
<td>Bambara</td>
<td>Ga</td>
<td>Kalmky</td>
<td>Mizo</td>
<td>Russian</td>
<td>Upper Sorbian</td>
</tr>
<tr>
<td>Bamilkele</td>
<td>Gaelic</td>
<td>Kamba</td>
<td>Macedonian</td>
<td>Rusyn</td>
<td>Ukrainian</td>
</tr>
<tr>
<td>Breton</td>
<td>Gagua</td>
<td>Kumaoni</td>
<td>Male</td>
<td>Ruanda</td>
<td>Urdu</td>
</tr>
<tr>
<td>Brahui</td>
<td>Galician</td>
<td>Komos</td>
<td>Malagasy</td>
<td>Russian</td>
<td>Uyghur</td>
</tr>
<tr>
<td>Brag Bhaska</td>
<td>Garshuni</td>
<td>Komos</td>
<td>Malakay</td>
<td>Russian</td>
<td>Upper Sorbian</td>
</tr>
<tr>
<td>Burmese</td>
<td>Garhwali</td>
<td>Kaniuri</td>
<td>Malinek</td>
<td>Sanskrit</td>
<td>Uzbek</td>
</tr>
<tr>
<td>Bashkir</td>
<td>Ge'ez</td>
<td>Kodagu</td>
<td>Malay</td>
<td>Sayisi</td>
<td>Vietnamese</td>
</tr>
<tr>
<td>Beti</td>
<td>Gilyak</td>
<td>Korean Old Hangul</td>
<td>Mindangda</td>
<td>Seokta</td>
<td>Wa</td>
</tr>
<tr>
<td>Catalan</td>
<td>Gurmuz</td>
<td>Konkani</td>
<td>Mongolian</td>
<td>Selkup</td>
<td>Wagdi</td>
</tr>
<tr>
<td>Cebuano</td>
<td>Gondi</td>
<td>Kikongo</td>
<td>Manipuri</td>
<td>Sango</td>
<td>West-Cree</td>
</tr>
<tr>
<td>Chechen</td>
<td>Greenlandic</td>
<td>Komi-Permyak</td>
<td>Maninka</td>
<td>Sango</td>
<td>Welsh</td>
</tr>
<tr>
<td>Chahage Gurage</td>
<td>Garo</td>
<td>Korean</td>
<td>Maxx Gaelic</td>
<td>Sibe</td>
<td>Wolof</td>
</tr>
<tr>
<td>Chattisgarhi</td>
<td>Guarani</td>
<td>Komi-Zyrian</td>
<td>Moksha</td>
<td>Sidamo</td>
<td>Tai Lue</td>
</tr>
<tr>
<td>Chichewa</td>
<td>Gujarati</td>
<td>Kpelle</td>
<td>Moldavian</td>
<td>Silte Gurge</td>
<td>Xhosa</td>
</tr>
<tr>
<td>Chukchi</td>
<td>Hailtian</td>
<td>Krio</td>
<td>Mon</td>
<td>Skolt Sami</td>
<td>Yakut</td>
</tr>
<tr>
<td>Chypewyan</td>
<td>Halam</td>
<td>Karalpak</td>
<td>Moroccan</td>
<td>Slovak</td>
<td>Yoruba</td>
</tr>
<tr>
<td>Cherokee</td>
<td>Harauti</td>
<td>Kareljan</td>
<td>Maori</td>
<td>Slavey</td>
<td>Y-Cree</td>
</tr>
<tr>
<td>Chuvash</td>
<td>Hausa</td>
<td>Karaim</td>
<td>Maithili</td>
<td>Slovenian</td>
<td>Yi Classic</td>
</tr>
<tr>
<td>Comorian</td>
<td>Hawaiiin</td>
<td>Karen</td>
<td>Maltese</td>
<td>Somali</td>
<td>Yi Modern</td>
</tr>
<tr>
<td>Coptic</td>
<td>Hammer-Banna</td>
<td>Koorete</td>
<td>Mundari</td>
<td>Samoan</td>
<td>Chinese Hong Kong</td>
</tr>
<tr>
<td>Cree</td>
<td>Hiligayon</td>
<td>Kashmiri</td>
<td>Naga-Assamese</td>
<td>Sana</td>
<td>Chinese Phonetic</td>
</tr>
<tr>
<td>Carrier</td>
<td>Hindi</td>
<td>Khasi</td>
<td>Naskapi</td>
<td>Nanai</td>
<td>Sinhalese</td>
</tr>
<tr>
<td>Crimean Tatar</td>
<td>High Mari</td>
<td>Kildin Sami</td>
<td>N-Kree</td>
<td>Soninke</td>
<td>Chinese Traditional</td>
</tr>
<tr>
<td>Church Slavonic</td>
<td>Hindko</td>
<td>Kui</td>
<td>Neebele</td>
<td>Sodo Gurge</td>
<td>Zande</td>
</tr>
<tr>
<td>Czech</td>
<td>Ho</td>
<td>Kulvi</td>
<td>Ndonga</td>
<td>Sotho</td>
<td>Zulu</td>
</tr>
<tr>
<td>Danish</td>
<td>Harari</td>
<td>Kumykyk</td>
<td>Nepali</td>
<td>Albanian</td>
<td>Zulu</td>
</tr>
<tr>
<td>Dargwa</td>
<td>Croatian</td>
<td>Kurukh</td>
<td>Newari</td>
<td>Serbian</td>
<td>Zulu</td>
</tr>
<tr>
<td>Woods Cree</td>
<td>Hungarian</td>
<td>Kurukh</td>
<td>Newari</td>
<td>Serbian</td>
<td>Zulu</td>
</tr>
<tr>
<td>German</td>
<td>Armenian</td>
<td>Kuy</td>
<td>Nagari</td>
<td>Saraiki</td>
<td>Zulu</td>
</tr>
</tbody>
</table>
Part III
LuaTeX-only font features

11 OpenType font feature files

An OpenType font feature file is a plain text file describing OpenType layout feature of a font in a human-readable format. The syntax of OpenType feature files is defined by Adobe\textsuperscript{8}.

Feature files can be used to add or customize OpenType features of a font on the fly without editing the font file itself.

Adding a new OpenType feature is as creating a plain text file defining the new feature and then loading it by passing its name or path to \texttt{FeatureFile}, then OpenType features defined in the file can be activated as usual.

For example, when adding one of the default features like \texttt{kern} or \texttt{liga}, no special activation is needed. On the other hand, an optional feature like \texttt{onum} or \texttt{smcp} will be activated when old style numbers or small capitals are activated, respectively. However, OpenType feature in the feature file can have any and that can be used to selectively activate the feature; for example defining a ligature feature called \texttt{mlig} and then activating it using \texttt{RawFeature} option without activating other ligatures in the font.

Figure 1 shows an example feature file. The first two lines set the script and language under which the defined features will be available, which the default language in both default and Latin scripts, respectively.

Then it defines a \texttt{liga} feature, which is a glyph substitution feature. The names starting with backslash are glyph names that is to be substituted and while the leading backslash is optional, it is used to escape glyph names when they interfere with preserved keywords. It should also be noted that glyph names are font specific and the same glyph can be named differently in different fonts.

Glyph positioning features like kerning can be defined in a similar way, but instead of the keyword \texttt{sub(stitute)} the keyword \texttt{pos(ition)} is used instead. Figure 1 shows an example of adding kerning between \texttt{AY} and \texttt{ay}\textsuperscript{9}.

Lines starting with \# are comments and will be ignored.

An OpenType feature file can have any number of features and can have a mix of substitution and positioning features, please refer to the full feature file specification for further documentation.

\textsuperscript{8}http://www.adobe.com/devnet/opentype/afrdko/topic_feature_file_syntax.html
\textsuperscript{9}The kerning is expressed in font design units which are fractions of em depending on the units per em value of the font, usually 1000 for PostScript fonts and 2048 for TrueType fonts.
languagesystem DFLT dflt;
languagesystem latn dflt;

# Ligatures
feature liga {
    sub \f \i by \fi;
    sub \f \l by \fl;
} liga;

# Kerning
feature kern {
    pos \u \y -2/zero.noslash/zero.noslash;
    pos \a \y -8/zero.noslash;
} kern;

---

Example 43: \LaTeX{}'s Mapping feature.

```
¡A small amount of—text!
```

\fontspec{Cochin}

```
"¡A small amount of—text!"
```

---

Part IV

Fonts and features with \LaTeX{}

12 \LaTeX{}-only font features

The features described here are available for any font selected by \fontspec.

12.1 Mapping

Mapping enables a \LaTeX{} text-mapping scheme, shown in Example 43.

Using the tex-text mapping is also equivalent to writing Ligatures=TeX. The use of the latter syntax is recommended for better compatibility with Lua\LaTeX{} documents.

12.2 Letter spacing

Letter spacing, or tracking, is the term given to adding (or subtracting) a small amount of horizontal space in between adjacent characters. It is specified with the LetterSpace, which takes a numeric argument, shown in Example 44.

The letter spacing parameter is a normalised additive factor (not a scaling factor); it is defined as a percentage of the font size. That is, for a 10 pt font, a letter spacing parameter of ‘1.0’ will add 0.1 pt between each letter.
Example 44: The LetterSpace feature.

\fontspec{Didot}\addfontfeature{LetterSpace=0.0}\USE TRACKING FOR DISPLAY CAPS TEXT \addfontfeature{LetterSpace=2.0}\USE TRACKING FOR DISPLAY CAPS TEXT

This functionality should not be used for lowercase text, which is spacing correctly to begin with, but it can be very useful, in small amounts, when setting small caps or all caps titles. Also see the OpenType Uppercase option of the Letters feature (Section 10.2 on page 23).

12.3 Different font technologies: AAT and ICU

\TeX supports two rendering technologies for typesetting, selected with the Renderer font feature. The first, AAT, is that provided (only) by Mac OS X itself. The second, ICU, is an open source OpenType interpreter. It provides much greater support for OpenType features, notably contextual arrangement, over AAT.

In general, this feature will not need to be explicitly called: for OpenType fonts, the ICU renderer is used automatically, and for AAT fonts, AAT is chosen by default. Some fonts, however, will contain font tables for both rendering technologies, such as the Hiragino Japanese fonts distributed with Mac OS X, and in these cases the choice may be required.

Among some other font features only available through a specific renderer, ICU provides for the Script and Language features, which allow different font behaviour for different alphabets and languages; see Section 10.18 on page 35 for the description of these features. Because these font features can change which features are able to be selected for the font instance, they are selected by fontspec before all others and will automatically and without warning select the ICU renderer.

12.4 Optical font sizes

Multiple Master fonts are parameterised over orthogonal font axes, allowing continuous selection along such features as weight, width, and optical size (see Section 14 on page 45 for further details). Whereas an OpenType font will have only a few separate optical sizes, a Multiple Master font’s optical size can be specified over a continuous range. Unfortunately, this flexibility makes it harder to create an automatic interface through \TeX, and the optical size for a Multiple Master font must always be specified explicitly.

\fontspec[OpticalSize=11]{Minion MM Roman}\ MM optical size test \ \ \ \fontspec[OpticalSize=47]{Minion MM Roman}\ MM optical size test

40
13  Mac OS X’s AAT fonts

Mac OS X’s font technology began life before the ubiquitous-OpenType era and revolved around the Apple-invented ‘AAT’ font format. This format had some advantages (and other disadvantages) but it never became widely popular in the font world.

Nonetheless, this is the font format that was first supported by \TeX (due to its pedigree on Mac OS X in the first place) and was the first font format supported by fontspec. A number of fonts distributed with Mac OS X are still in the AAT format, such as ‘Skia’. Documents that use these fonts should be compiled with \TeX using the xdv2pdf driver, as opposed to the default xdvipdfmx. E.g.,

\texttt{xelatex -output-driver="xdv2pdf" filename.tex}

Mac OS X also supports Multiple Master fonts, which are discussed in Section 14.

13.1  Ligatures

Ligatures refer to the replacement of two separate characters with a specially drawn glyph for functional or aesthetic reasons. For AAT fonts, you may choose from any combination of Required, Common, Rare (or Discretionary), Logos, Rebus, Diphthong, Squared, AbbrevSquared, and Icelandic.

Some other Apple AAT fonts have those ‘Rare’ ligatures contained in the Icelandic feature. Notice also that the old \TeX trick of splitting up a ligature with an empty brace pair does not work in \TeX; you must use a 0 pt kern or \hbox (e.g., \null) to split the characters up if you do not want a ligature to be performed (the usual examples for when this might be desired are words like ‘shelffull’).

13.2  Letters

The Letters feature specifies how the letters in the current font will look. For AAT fonts, you may choose from Normal, Uppercase, Lowercase, SmallCaps, and InitialCaps.

13.3  Numbers

The Numbers feature defines how numbers will look in the selected font. For AAT fonts, they may be a combination of Lining or OldStyle and Proportional or Monospaced (the latter is good for tabular material). The synonyms Uppercase and Lowercase are equivalent to Lining and OldStyle, respectively. The differences have been shown previously in Section 7.2 on page 13.
Example 45: Contextual glyph for the beginnings and ends of words.

\newfontface\fancy [Contextuals={(WordInitial,WordFinal)}] (Hoefer Text Italic) \fancy where is all the vegemite

Example 46: A contextual feature for the ‘long s’ can be convenient as the character does not need to be marked up explicitly.

\fontspec[Contextuals=Inner]{Hoefler Text} ‘Inner’ swashes can \emph{sometimes} contain the archaic long s.

13.4 Contextuals

This feature refers to glyph substitution that vary by their position; things like contextual swashes are implemented here. The options for AAT fonts are WordInitial, WordFinal (Example 45), LineInitial, LineFinal, and Inner (Example 46, also called ‘non-final’ sometimes). As non-exclusive selectors, like the ligatures, you can turn them off by prefixing their name with No.

13.5 Vertical position

The VerticalPosition feature is used to access things like subscript (Inferior) and superscript (Superior) numbers and letters (and a small amount of punctuation, sometimes). The Ordinal option is (supposed to be) contextually sensitive to only raise characters that appear directly after a number. These are shown in Example 47.

The realscripts package (also loaded by \texttt{xltxtra}) redefines the \texttt{textsubscript} and \texttt{textsuperscript} commands to use the above font features, including for use in footnote labels.

Example 47: Vertical position for AAT fonts.

\fontspec{Skia} Normal \fontspec[VerticalPosition=Superior]{Skia} Superior \fontspec[VerticalPosition=Inferior]{Skia} Inferior \fontspec[VerticalPosition=Ordinal]{Skia} 1st 2nd 3rd 4th 0th 8abcde
Example 48: Fractions in AAT fonts. The "2/0" glyph is the ‘fraction slash’ that may be typed in Mac OS X with opt+shift+1; not shown literally here due to font constraints.

<table>
<thead>
<tr>
<th>Fraction</th>
<th>AAT Code</th>
<th>Fontspec Command</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2 5/6</td>
<td>1/2 5/6</td>
<td>\fontspec{Skia}</td>
<td>fraction slash</td>
</tr>
<tr>
<td>1/2 5/6</td>
<td>1/2 5/6</td>
<td>\fontspec{Skia}</td>
<td>regular slash</td>
</tr>
<tr>
<td>13579/24680</td>
<td>13579/24680</td>
<td>\fontspec{Skia}</td>
<td>fraction slash</td>
</tr>
<tr>
<td>13579/24680</td>
<td>13579/24680</td>
<td>\fontspec{Skia}</td>
<td>regular slash</td>
</tr>
</tbody>
</table>

Example 49: Alternate design of pre-composed fractions.

\fontspec{Hiragino Maru Gothic Pro}
\addfontfeature{Fractions=Alternate}

<table>
<thead>
<tr>
<th>Fraction</th>
<th>AAT Code</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2 1/4 5/6</td>
<td>1/2 1/4 5/6</td>
<td>\quad 1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 % fraction slash</td>
</tr>
<tr>
<td>1/2 1/4 5/6</td>
<td>1/2 1/4 5/6</td>
<td>\quad 1/2 \quad 1/4 \quad 5/6 \quad 13579/24680 % regular slash</td>
</tr>
</tbody>
</table>

13.6 Fractions

Many fonts come with the capability to typeset various forms of fractional material. This is accessed in \fontspec with the Fractions feature, which may be turned On or Off in both AAT and OpenType fonts.

In AAT fonts, the ‘fraction slash’ or solidus character, is to be used to create fractions. When Fractions are turned On, then only pre-drawn fractions will be used. See Example 48.

Using the Diagonal option (AAT only), the font will attempt to create the fraction from superscript and subscript characters.

Some (Asian fonts predominantly) also provide for the Alternate feature shown in Example 49.

13.7 Variants

The Variant feature takes a single numerical input for choosing different alphabetic shapes. Don’t mind my fancy Example 50 :) I’m just looping through the nine (!) variants of Zapfino.

See Section 15 on page 46 for a way to assign names to variants, which should be done on a per-font basis.

13.8 Alternates

Selection of Alternates again must be done numerically; see Example 51. See Section 15 on page 46 for a way to assign names to alternates, which should be done on a per-font basis.
Example 50: Nine variants of Zapfino.

\newcounter{var}\newcounter{trans}
\whiledo{value(var)<9}{%
  \stepcounter{trans}%
  \edef\1{%
    \noexpand\fontspec[Variant=thevar,\thetrans\thetrans\thetrans\thetrans\{Zapfino}\%\makebox[0.75\width][d]{}
    \stepcounter{var}}%
}

Example 51: Alternate shape selection must be numerical.

\fontsizepec[Alternate=/zero.noslash]{Hoefler Text Italic}
Sphinx Of Black Quartz, \textsc{Judge My Vow} \\ 
\fontsizepec[Alternate=1]{Hoefler Text Italic}
Sphinx Of Black Quartz, \textsc{Judge My Vow}

13.9 Style

The options of the Style feature are defined in AAT as one of the following: Display, Engraved, IlluminatedCaps, Italic, Ruby,\footnote{Ruby refers to a small optical size, used in Japanese typography for annotations.} TallCaps, or TitlingCaps.

Typical examples for these features are shown in Section 10.10.

13.10 CJK shape

There have been many standards for how CJK ideographic glyphs are ‘supposed’ to look. Some fonts will contain many alternate glyphs in order to be able to display these glyphs correctly in whichever form is appropriate. Both AAT and OpenType fonts support the following CJKShape options: Traditional, Simplified, JIS1978, JIS1983, JIS1990, and Expert. OpenType also supports the NLC option.

13.11 Character width

See Section 10.16 on page 32 for relevant examples; the features are the same between OpenType and AAT fonts. AAT also allows CharacterWidth=Default to return to the original font settings.

13.12 Vertical typesetting

TODO: improve!

\TeX{} provides for vertical typesetting simply with the ability to rotate the individual glyphs as a font is used for typesetting, as shown in Example 52.
Example 52: Vertical typesetting.

No actual provision is made for typesetting top-to-bottom languages; for an example of how to do this, see the vertical Chinese example provided in the \TeX documentation.

13.13 Diacritics

Diacritics are marks, such as the acute accent or the tilde, applied to letters; they usually indicate a change in pronunciation. In Arabic scripts, diacritics are used to indicate vowels. You may either choose to Show, Hide or Decompose them in \aat fonts. The Hide option is for scripts such as Arabic which may be displayed either with or without vowel markings. E.g., \texttt{\fontspec[Diacritics=Hide]{...}}

Some older fonts distributed with Mac OS X included 'Ø' etc. as shorthand for writing 'Ø' under the label of the Diacritics feature. If you come across such fonts, you’ll want to turn this feature off (imagine typing hello/goodbye and getting ‘helløgoodbye’ instead!) by decomposing the two characters in the diacritic into the ones you actually want. I recommend using the proper \TeX input conventions for obtaining such characters instead.

13.14 Annotation

Various Asian fonts are equipped with a more extensive range of numbers and numerals in different forms. These are accessed through the Annotation feature (see Example 53) with the following options: Off, Box, RoundedBox, Circle, BlackCircle, Parenthesis, Period, RomanNumerals, Diamond, BlackSquare, BlackRoundSquare, and DoubleCircle.

14 \aat & Multiple Master font axes

Multiple Master and \aat font specifications both provide continuous variation along font parameters. For example, they don’t have just regular and bold weights, they can have any bold weight you like between the two extremes. Note these features can only be used when your document is compiled using the xdvi2pdf driver for Mac OS X.

Weight, Width, and OpticalSize are supported by this package. Skia, which is distributed with Mac OS X, has two of these variable parameters, allowing for the
Example 53: Various annotation forms.

```
\fontspec{Hei Regular}
1 2 3 4 5 6 7 8 9
\fontspec[Annotation=Circle]{Hei Regular}
1 2 3 4 5 6 7 8 9
\fontspec[Annotation=Parenthesis]{Hei Regular}
1 2 3 4 5 6 7 8 9
\fontspec[Annotation=Period]{Hei Regular}
1 2 3 4 5 6 7 8 9
```

Example 54: Continuously variable font parameters. These fonts are unfortunately quite rare.

```
\fontspec[Weight=/zero.noslash.5,Width=3]{Skia}
Really light and extended Skia
\fontspec[Weight=2,Width=/zero.noslash.5]{Skia}
Really fat and condensed Skia
```

demonstration in Example 54. Variations along a multiple master font’s optical size axis has been shown previously in Section 8.6 on page 19.

Part V

Programming interface

This is the beginning of some work to provide some hooks that use \fontspec for various macro programming purposes.

15 Defining new features

This package cannot hope to contain every possible font feature. Three commands are provided for selecting font features that are not provided for out of the box. If you are using them a lot, chances are I’ve left something out, so please let me know.

```
\newAATfeature{\langle feature \rangle}{\langle option \rangle}{\langle feature code \rangle}{\langle selector code \rangle}
```

New AAT features may be created with this command:

Use the \texttt{Xe\TeX} file \texttt{AAT-info.tex} to obtain the code numbers. See Example 55.

```
\newICUfeature{\langle feature \rangle}{\langle option \rangle}{\langle feature tag \rangle}
```

New OpenType features may be created with this command:

The synonym \texttt{\newopentypefeature} is provided for Lua\texttt{\LaTeX} users.

```
\newICUfeature{\langle feature \rangle}{\langle option \rangle}{\langle feature tag \rangle}
```

Here’s what it would look like in practise:

```
\newopentypefeature{Style}{NoLocalForms}{-locl}
```

In case the above commands do not accommodate the desired font feature
Example 55: Assigning new \texttt{aat} features.

This is \LaTeX\textipa by Jonathan Kew.

\begin{verbatim}
  \newAATfeature{Alternate}{HoeflerSwash}{17}{1}
  \fontspec[Alternate=HoeflerSwash]{Hoefler Text Italic}
\end{verbatim}

Example 56: Assigning new arbitrary features.

\begin{verbatim}
  \newfontfeature{AvoidD}{Special=Avoid d-collisions}
  \newfontfeature{NoAvoidD}{Special=!Avoid d-collisions}
  \fontspec[AvoidD,Variant=1]{Zapfino}
  sockdolager rubdown
  \fontspec[NoAvoidD,Variant=1]{Zapfino}
  sockdolager rubdown
\end{verbatim}

(Perhaps a new \LaTeX\textipa feature that \texttt{fontspec} hasn’t been updated to support), a command is provided to pass arbitrary input into the font selection string:

\begin{verbatim}
  \newfontfeature{⟨name⟩}{⟨input string⟩}
\end{verbatim}

For example, Zapfino contains the feature ‘Avoid d-collisions’. To access it with this package, you could do something like that shown in Example 56.

The advantage to using the \texttt{\newAATfeature} and \texttt{\newICUfeature} commands instead of \texttt{\newfontfeature} is that they check if the selected font actually contains the desired font feature at load time. By contrast, \texttt{\newfontfeature} will not give a warning for improper input.

16 Going behind \texttt{fontspec}'s back

Expert users may wish not to use \texttt{fontspec}'s feature handling at all, while still taking advantage of its \LaTeX\textipa font selection conveniences. The \texttt{RawFeature} font feature allows literal \LaTeX\textipa font feature selection when you happen to have the OpenType feature tag memorised.

Multiple features can either be included in a single declaration:

\begin{verbatim}
  [RawFeature=+smcp;+onum]
\end{verbatim}

or with multiple declarations:

\begin{verbatim}
  [RawFeature=+smcp, RawFeature=+onum]
\end{verbatim}

Example 57: Using raw font features directly.

\begin{verbatim}
  \fontspec[RawFeature=+smcp]{\TeX\textipa Gyre Pagella}
  Pagella small caps
\end{verbatim}
17 Renaming existing features & options

If you don’t like the name of a particular font feature, it may be aliased to another with the \aliasfontfeature{⟨existing name⟩}{⟨new name⟩} command, such as shown in Example 58.

Spaces in feature (and option names, see below) are allowed. (You may have noticed this already in the lists of OpenType scripts and languages).

If you wish to change the name of a font feature option, it can be aliased to another with the command \aliasfontfeatureoption{⟨font feature⟩}{⟨existing name⟩}{⟨new name⟩}, such as shown in Example 59.

This example demonstrates an important point: when aliasing the feature options, the original feature name must be used when declaring to which feature the option belongs.

Only feature options that exist as sets of fixed strings may be altered in this way. That is, Proportional can be aliased to Prop in the Letters feature, but 55099BB cannot be substituted for Purple in a Color specification. For this type of thing, the \newfontfeature command should be used to declare a new, e.g., PurpleColor feature:

\newfontfeature{PurpleColor}{color=55099BB}

Except that this example was written before support for named colours was implemented. But you get the idea.

18 Programming details

In some cases, it is useful to know what the \TeX font family of a specific fontspec font is. After a \fontspec-like command, this is stored inside the \l_fontspec_family_tl macro. Otherwise, \TeX’s own \f@family macro can be useful here, too. The raw \TeX font that is defined is stored temporarily in \l_fontspec_font.
The following commands in expl3 syntax may be used for writing codes that interface with fontspec-loaded fonts. All of the following conditionals also exist with \text{T} and \text{F} as well as \text{TF} suffixes.

\fontspec_if_fontspec_font:TF Test whether the currently selected font has been loaded by fontspec.

\fontspec_if_aat_feature:nnTF Test whether the currently selected font contains the \text{aat} feature (#1,#2).

\fontspec_if_opentype:TF Test whether the currently selected font is an OpenType font. Always true for \LaTeX fonts.

\fontspec_if_feature:nTF Test whether the currently selected font contains the raw OpenType feature #1. E.g.: \fontspec_if_feature:nTF \{pnum\} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_feature:nnnTF Test whether the currently selected font with raw OpenType script tag #1 and raw OpenType language tag #2 contains the raw OpenType feature tag #3. E.g.: \fontspec_if_feature:nnnTF \{latn\} \{ROM\} \{pnum\} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_script:nTF Test whether the currently selected font contains the raw OpenType script #1. E.g.: \fontspec_if_script:nTF \{latn\} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_language:nTF Test whether the currently selected font contains the raw OpenType language tag #1. E.g.: \fontspec_if_language:nTF \{ROM\} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_language:nnTF Test whether the currently selected font contains the raw OpenType language tag #2 in script #1. E.g.: \fontspec_if_language:nnTF \{cyr\} \{SRB\} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_current_script:nTF Test whether the currently loaded font is using the specified raw OpenType script tag #1.

\fontspec_if_current_language:nTF Test whether the currently loaded font is using the specified raw OpenType language tag #1.

\fontspec_set_family:Nnn \#1 : \LaTeX family
\#2 : fontspec features
\#3 : font name
Defines a new NFSS family from given \text{features} and \text{font}, and stores the family name in the variable \text{family}. This font family can then be selected with standard \LaTeX commands \text{\fontfamily{\family}\selectfont}. See the standard fontspec user commands for applications of this function.

\fontspec_set_fontface:NNnn \#1 : primitive font
\#2 : \LaTeX family
\#3 : fontspec features
\#4 : font name
Variant of the above in which the primitive \TeX{} font command is stored in the variable \texttt{⟨primitive font⟩}. If a family is loaded (with bold and italic shapes) the primitive font command will only select the regular face. This feature is designed for \LaTeX{} programmers who need to perform subsequent font-related tests on the \texttt{⟨primitive font⟩}.

Part VI

The patching/improvement of \LaTeX{} 2ε and other packages

Derived originally from xltxta, this package contains patches to various \LaTeX{} components and third-party packages to improve the default behaviour.

19 Inner emphasis

\texttt{fixltt2ε}'s method for checking for “inner” emphasis is a little fragile in \Xe\TeX{}, because font slant information might be missing from the font. Therefore, we use \LaTeX{}'s NFSS information, which is more likely to be correct.

20 Unicode footnote symbols

By default \LaTeX{} defines symbolic footnote characters in terms of commands that don't resolve well; better results can be achieved by using specific Unicode characters or proper LICRs with the \texttt{xunicode} package.

This problem has been solved by loading the \texttt{fixltt2ε} package.

21 Verbatim

Many verbatim mechanisms assume the existence of a ‘visible space’ character that exists in the \texttt{ascii} space slot of the typewriter font. This character is known in Unicode as \texttt{U+2434: box open}, which looks like this: ‘␣’.

When a Unicode typewriter font is used, \LaTeX{} no longer prints visible spaces for the \texttt{verbatim*} environment and \texttt{\verb*} command. This problem is fixed by using the correct Unicode glyph, and the following packages are patched to do the same: listings, fancyverb, moreverb, and verbatim.

In the case that the typewriter font does not contain ‘␣’, the Latin Modern Mono font is used as a fallback.

22 Discretionary hyphenation: \texttt{\-}

\LaTeX{} defines the macro \texttt{\-} to insert discretionary hyphenation points. However, it is hard-coded in \LaTeX{} to use the hyphen - character. Since fontspec makes it
easy to change the hyphenation character on a per font basis, it would be nice if \-adjusted automatically — and now it does.

23 Commands for old-style and lining numbers

\oldstylenums \liningnums  \LaTeX's definition of \oldstylenums relies on strange font encodings. We provide a fontspec-compatible alternative and while we're at it also throw in the reverse option as well. Use \oldstylenums{⟨text⟩} to explicitly use old-style (or lowercase) numbers in ⟨text⟩, and the reverse for \liningnums{⟨text⟩}.
Part VII

fontspec.sty and friends

Herein lie the implementation details of this package. Welcome! It was my first.

24 ‘Header’ code

We will eventually load the correct version of the code according to which engine we’re running. As we’ll see later, there are some minor differences between what we have to do in Xe\TeX and Lua\TeX.

\begin{verbatim}
1 \langle fontspec&!xetex&!\texttt{\textregistered}luatex \rangle
\end{verbatim}

But for now, this is the shared code.

\begin{verbatim}
2 \RequirePackage{expl3}[2011/09/05]
3 \RequirePackage{xparse}
4 \ExplSyntaxOn
5 \msg_new:nnn {fontspec} {cannot-use-pdftex}
6 {
7 \msg_\textregistered:nnn {fontspec} {cannot-use-pdftex}
8 \textregistered:nnn {fontspec} {cannot-use-pdftex}
9 You must change your typesetting engine to, e.g., "xelatex" or "\textregistered\textregistered\textregistered\textregistered\textregistered latex"
10 instead of "plain" "\textregistered\textregistered\textregistered\textregistered\textregistered latex".
11}
12 \xetex_if_engine:F {
13 \luatex_if_engine:TF {
14 \RequirePackage{luaotfload}
15 \RequireLuaModule{fontspec}
16 }
17 \msg_fatal:nn {fontspec} {cannot-use-pdftex}
18 }
19 }
20 }
\end{verbatim}

24.1 expl3 tools

24.2 Bits and pieces

Conditionals
\begin{verbatim}
21 \bool_new:N \l_fontspec_firsttime_bool
22 \bool_new:N \l_fontspec_nobf_bool
23 \bool_new:N \l_fontspec_noit_bool
24 \bool_new:N \l_fontspec_nosc_bool
25 \bool_new:N \l_fontspec_tfm_bool
26 \bool_new:N \l_fontspec_atsui_bool
27 \bool_new:N \l_fontspec_icu_bool
\end{verbatim}
\bool_new:N \l_fontspec_mm_bool
\bool_new:N \l_fontspec_graphite_bool

For dealing with legacy maths
\bool_new:N \g_fontspec_math_euler_bool
\bool_new:N \g_fontspec_math_lucida_bool
\bool_new:N \g_fontspec_package_euler_loaded_bool

For package options:
\bool_new:N \g_fontspec_cfg_bool
\bool_new:N \g_fontspec_math_bool

Counters
\int_new:N \l_fontspec_script_int
\int_new:N \l_fontspec_language_int
\int_new:N \l_fontspec_strnum_int

Other variables
\fp_new:N \l_fontspec_tmpa_fp
\fp_new:N \l_fontspec_tmpb_fp
\dim_new:N \l_fontspec_tmpa_dim
\dim_new:N \l_fontspec_tmpb_dim
\dim_new:N \l_fontspec_tmpc_dim
\tl_set:Nx \c_colon_str { \tl_to_str:N : }
\cs_set:Npn \use_v:nnnnn #1#2#3#4#5 {#5}
\cs_set:Npn \use_iv:nnnnn #1#2#3#4#5 {#4}

Need these:
\cs_generate_variant:Nn \str_if_eq:nnTF {nv}
\cs_generate_variant:Nn \int_set:Nn {Nv}
\cs_generate_variant:Nn \tl_gset:Nn {cV}
\keys_set:nn {nx}
\keys_set_known:nnN {nx}
\_int_mult_truncate:Nn

Missing in expl3,IMO.
\cs_new:Nn \_int_mult_truncate:Nn
\{ %
\int_set:Nn \l_int_mult_truncate:Nn %
\{ \dim_eval:w \dim_eval_end: %
\}
\}

24.3 Error/warning/info messages

Shorthands for messages:
\cs_new:Npn \fontspec_error:n { \msg_error:nn \{ \fontspec \} }
\cs_new:Npn \fontspec_error:nx { \msg_error:nx \{ \fontspec \} }
\cs_new:Npn \fontspec_warning:n { \msg_warning:nn \{ \fontspec \} }
\cs_new:Npn \fontspec_warning:nx { \msg_warning:nx \{ \fontspec \} }
\cs_new:Npn \fontspec_info:n { \msg_info:n \{ \fontspec \} }
\cs_new:Npn \fontspec_info:nx { \msg_info:nx \{ \fontspec \} }
Errors:

\msg_new:nnn {fontspec} {no-size-info}

\msg_new:nnn {fontspec} {font-not-found}

\msg_new:nnnn {fontspec} {rename-feature-not-exist}

\msg_new:nnn {fontspec} {no-glyph}

\msg_new:nnnn {fontspec} {euler-too-late}

\msg_new:nnnn {fontspec} {no-xcolor}

\msg_new:nnnn {fontspec} {unknown-color-model}
Sorry, I can't do anything to help. Please report this error to my developer with a minimal example that causes the problem.

Warnings:

\msg_new:nnn {fontspec} {addfontfeatures-ignored} \string\addfontfeature \string s ignored; it cannot be used with a font that wasn't selected by fontspec.

\msg_new:nnn {fontspec} {feature-option-overwrite} Option '#2' of font feature '#1' overwritten.

\msg_new:nnn {fontspec} {script-not-exist-latin} Font '{\l_fontspec_fontname_tl}' does not contain script '#1'. 'Latin' script used instead.

\msg_new:nnn {fontspec} {script-not-exist} Font '{\l_fontspec_fontname_tl}' does not contain script '#1'.

\msg_new:nnn {fontspec} {aat-feature-not-exist} '{\l_keys_key_tl}={\l_keys_value_tl}' feature not supported for AAT font '{\l_fontspec_fontname_tl}'.

\msg_new:nnn {fontspec} {aat-feature-not-exist-in-font} AAT feature '{\l_keys_key_tl}={\l_keys_value_tl}' (#1) not available in font '{\l_fontspec_fontname_tl}'.

\msg_new:nnn {fontspec} {icu-feature-not-exist} '{\l_keys_key_tl}={\l_keys_value_tl}' feature not supported for ICU font '{\l_fontspec_fontname_tl}'.

\msg_new:nnn {fontspec} {icu-feature-not-exist-in-font} OpenType feature '{\l_keys_key_tl}={\l_keys_value_tl}' (#1) not available for font '{\l_fontspec_fontname_tl}' with script '{\l_fontspec_script_name_tl}' and language '{\l_fontspec_lang_name_tl}'.

\msg_new:nnn {fontspec} {no-opticals} '{\l_fontspec_fontname_tl}' doesn't appear to have an Optical Size axis.

\msg_new:nnn {fontspec} {language-not-exist} Language '#1' not available for font '{\l_fontspec_fontname_tl}'
with `\l_fontspec_script_name_tl'.\"
'Default' language used instead.
\}
\msg_new:nnn {fontspec} {only-xetex-feature}
\{\nIgnored XeTeX only feature: '#1'.
\}
\msg_new:nnn {fontspec} {only-luatex-feature}
\{\nIgnored LuaTeX only feature: '#1'.
\}
\msg_new:nnn {fontspec} {no-mapping}
\{\nInput mapping not (yet?) supported in LuaTeX.
\}
\msg_new:nnn {fontspec} {no-mapping-ligtex}
\{\nUse "Ligatures=TeX" instead of "Mapping= tex-text".
\}
\msg_new:nnn {fontspec} {cm-default-obsolete}
\{\nThe "cm-default" package option is obsolete.
\}
\msg_new:nnn {fontspec} {fakebold-only-xetex}
\{\nThe "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.
\}
\msg_new:nnn {fontspec} {defining-font}
\{\nFont family\"\l_fontspec_family_tl\" created for font '#2' with options \[g_fontspec_default_fontopts_tl #1\].\"
\}
\msg_new:nnn {fontspec} {no-font-shape}
\{\nCould not resolve font '#1' (it probably doesn't exist).
\}
\msg_new:nnn {fontspec} {set-scale}
\{\n\l_fontspec_fontname_tl\space scale \"=\" \l_fontspec_scale_tl.
\}
\msg_new:nnn {fontspec} {setup-math}
\{\nAdjusting the maths setup (use [no-math] to avoid this).
\}
\msg_new:nnn {fontspec} {no-scripts}
Font \_l\_fontspec\_fontname\_tt\space does not contain any OpenType ‘Script’ information.
\msg_new:nnn {fontspec} {opa-twice}

Opacity set twice, in both Colour and Opacity.
Using specification “Opacity=#1”.

\msg_new:nnn {fontspec} {opa-twice-col}

Opacity set twice, in both Opacity and Colour.
Using an opacity specification in hex of “#1/FF”.

\msg_new:nnn {fontspec} {bad-colour}

Bad colour declaration “#1”.
Colour must be one of:
- a named xcolor colour
- a six-digit hex colour RRGGBB
- an eight-digit hex colour RRGGBBTT with opacity

\section*{24.4 Option processing}
\DeclareOption{cm-default}{\fontspec_warning:n {cm-default-obsolete}}
\DeclareOption{math}{\bool_set_true:N \g_fontspec_math_bool}
\DeclareOption{no-math}{\bool_set_false:N \g_fontspec_math_bool}
\DeclareOption{config}{\bool_set_true:N \g_fontspec_cfg_bool}
\DeclareOption{no-config}{\bool_set_false:N \g_fontspec_cfg_bool}
\DeclareOption{quiet}{\msg_redirect_module:nnn { fontspec } { warning } { info }}
\DeclareOption{silent}{\msg_redirect_module:nnn { fontspec } { warning } { none }}
\ExecuteOptions{config,math}
\ProcessOptions*

\section*{24.5 Packages}
New for Lua\TeX, we load a new package called ‘fontspec-patches’ designed to incorporate the hidden but useful parts of the old \texttt{xltexra} package.
\RequirePackage{fontspec-patches}
\luatex_if_engine:T { \RequirePackage{fontspec-luatex} \endinput }
\xetex_if_engine:T { \RequirePackage{fontspec-xetex} \endinput }
\fontspec!\xetexxx\&\luatex
25 The main package code

That was the driver, and now the fun starts.

254 \fontspec \ExplSyntaxOn

25.1 Encodings

Frank Mittelbach has recommended using the ‘EUx’ family of font encodings to experiment with Unicode. Now that \TeX\ can find fonts in the \texttt{texmf} tree, the Latin Modern OpenType fonts can be used as the defaults. See the \texttt{euenc} collection of files for how this is implemented.

\begin{verbatim}
\tl_set:Nn \g_fontspec_encoding_tl {EU1}
\tl_set:Nn \g_fontspec_encoding_tl {EU2}
\tl_set:Nn \rmdefault {lmr}
\tl_set:Nn \sfdefault {lmss}
\tl_set:Nn \ttdefault {lmtt}
\RequirePackage[\g_fontspec_encoding_tl]{fontenc}
\tl_set_eq:NN \UTFencname \g_fontspec_encoding_tl % for xunicode
\end{verbatim}

Dealing with a couple of the problems introduced by babel:

\begin{verbatim}
\tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
\tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
\tl_put_right:Nn \document {
\tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
\tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
}
\end{verbatim}

That latin encoding definition is repeated to suppress font warnings. Something to do with \texttt{\select@language} ending up in the \texttt{.aux} file which is read at the beginning of the document.

\texttt{xunicode} Now we load \texttt{xunicode}, working around its internal \TeX\ check when under \LaTeX.

\begin{verbatim}
\RequirePackage{xunicode}
\end{verbatim}

25.2 User commands

This section contains the definitions of the commands detailed in the user documentation. Only the ‘top level’ definitions of the commands are contained herein; they all use or define macros which are defined or used later on in Section 25.5 on page 69.
25.2.1 Font selection

\fontspec\ This is the main command of the package that selects fonts with various features. It takes two arguments: the font name and the optional requested features of that font. Then this new font family is selected.

276 \DeclareDocumentCommand \fontspec { O{} m } {
277 \fontencoding {\g_fontspec_encoding_tl}
278 \fontspec_set_family:NNn \f@family {#1}{#2}
279 \selectfont
280 \ignorespaces
281 }

\setmainfont\ The following three macros perform equivalent operations setting the default font for a particular family: ‘roman’, sans serif, or typewriter (monospaced). I end them with \normalfont so that if they’re used in the document, the change registers immediately.

282 \DeclareDocumentCommand \setmainfont { O{} m } {
283 \fontspec_set_family:NNn \rmdefault {#1}{#2}
284 \normalfont
285 }
286 \DeclareDocumentCommand \setsansfont { O{} m } {
287 \fontspec_set_family:NNn \sfdefault {#1}{#2}
288 \normalfont
289 }
290 \DeclareDocumentCommand \setmonofont { O{} m } {
291 \fontspec_set_family:NNn \ttdefault {#1}{#2}
292 \normalfont
293 }

\setromanfont\ This is the old name for \setmainfont, retained for backwards compatibility.

294 \cs_set_eq:NN \setromanfont \setmainfont

\setmathrm\ These commands are analogous to \setromanfont and others, but for selecting the font used for \mathrm, etc. They can only be used in the preamble of the document. \setboldmathrm is used for specifying which fonts should be used in \boldmath.

295 \tl_new:N \g_fontspec_mathrm_tl
296 \tl_new:N \g_fontspec_bfmathrm_tl
297 \tl_new:N \g_fontspec_mathsf_tl
298 \tl_new:N \g_fontspec_mathtt_tl
299 \DeclareDocumentCommand \setmathrm { O{} m } {
300 \fontspec_set_family:Nn \g_fontspec_mathrm_tl {#1}{#2}
301 }
302 \DeclareDocumentCommand \setboldmathrm { O{} m } {
303 \fontspec_set_family:Nn \g_fontspec_bfmathrm_tl {#1}{#2}
304 }
305 \DeclareDocumentCommand \setmathsf { O{} m } {
306 \fontspec_set_family:Nn \g_fontspec_mathsf_tl {#1}{#2}
307 }
308 \DeclareDocumentCommand \setmathtt { O{} m } {
309 \fontspec_set_family:Nn \g_fontspec_mathtt_tl {#1}{#2}
310 }

59
If the commands above are not executed, then \texttt{\rmdefault} (etc.) will be used.

\tl_set:Nn \g_fontspec_mathrm_tl {\rmdefault}
\tl_set:Nn \g_fontspec_mathsfr_tl {\sffamily}
\tl_set:Nn \g_fontspec_mathssf_tl {\ttdefault}

\newfontfamily This macro takes the arguments of \fontspec with a prepended \texttt{(instance cmd)} (code for middle optional argument generated by Scott Pakin’s newcommand.py). This command is used when a specific font instance needs to be referred to repetitively (e.g., in a section heading) since continuously calling \fontspec\_select:nn is inefficient because it must parse the option arguments every time.

\fontspec\_select:nn defines a font family and saves its name in %l_fontspec\_family\_tl. This family is then used in a typical NFSS \fontfamily declaration, saved in the macro name specified.

\DeclareDocumentCommand \newfontfamily { m O{} m } {
\fontspec\_select:nn{#2}{#3}
\use:x {
\exp_not:N \DeclareRobustCommand \exp_not:N #1 {
\exp_not:N \fontencoding {\g_fontspec\_encoding\_tl}
\exp_not:N \fontfamily {\l_fontspec\_family\_tl} \exp_not:N \selectfont
}
}
}

\newfontface uses an undocumented feature of the BoldFont feature; if its argument is empty (=i.e., BoldFont=()), then no bold font is searched for.

\DeclareDocumentCommand \newfontface { m O{} m } {
\newfontfamily #1 [ BoldFont={},ItalicFont={},SmallCapsFont={},#2 ] {#3}
}

\NewDocumentCommand \defaultfontfeatures \defaultfontfeatures This macro takes one argument that consists of all of feature options that will be applied by default to all subsequent \fontspec, et al., commands. It stores its value in %g_fontspec\_default\_fontopts\_tl (initialised empty), which is concatenated with the individual macro choices in the [...] macro.

\tl_new:N \g_fontspec\_default\_fontopts\_tl
\DeclareDocumentCommand \defaultfontfeatures {m} {
\tl_set:Nn \g_fontspec\_default\_fontopts\_tl {#1,}
}

\addfontfeatures In order to be able to extend the feature selection of a given font, two things need to be known: the currently selected features, and the currently selected font. Every time a font family is created, this information is saved inside a control sequence with the name of the font family itself.
This macro extracts this information, then appends the requested font features
to add to the already existing ones, and calls the font again with the top level
\fontspec command.

The default options are not applied (which is why \g_fontspec_default_fontopts_tl
is emptied inside the group; this is allowed as $l_fontspec_family_tl$ is globally
defined in $\fontspec_select:nn$), so this means that the only added features to
the font are strictly those specified by this command.

$\addfontfeatures$ is defined as an alias, as I found that I often typed this instead
when adding only a single font feature.

25.2.3 Defining new font features

$\newfontfeature$ takes two arguments: the name of the feature tag by which to
reference it, and the string that is used to select the font feature.

NewAATfeature

This command assigns a new AAT feature by its code (#2,#3) to a new name (#1).
Better than $\newfontfeature$ because it checks if the feature exists in the font it's
being used for.
\newfontscript

Mostly used internally, but also possibly useful for users, to define new OpenType ‘scripts’, mapping logical names to OpenType script tags. Iterates through the scripts

\aliasfontfeature

User commands for renaming font features and font feature options.

\aliasfontfeatureoption

366 \fontspec_define_feature_option:nnnnn[#1][#2][#3][#4]{}
367

\newICUfeature This command assigns a new OpenType feature by its abbreviation (#2) to a new name (#1). Better than \newfontfeature because it checks if the feature exists in the font it’s being used for.

368 \DeclareDocumentCommand \newICUfeature {mmm} {
369 \keys_if_exist:nnF { fontspec / options } (#1) {
370 \fontspec_define_font_feature:n{#1} }
371 \keys_if_choice_exist:nnnT {fontspec} {#1} {#2} {
373 \fontspec_define_feature_option:nnnnn[#1][#2][#3][#4]{}
374 }
375 \cs_set_eq:NN \newopentypefeature \newICUfeature

\aliasfontfeature

\aliasfontfeatureoption

376 \DeclareDocumentCommand \aliasfontfeature {mm} {
377 \keys_if_exist:nnTF {fontspec} {#1} {
378 \keys_define:nn {fontspec} { #2 .code:n = { \keys_set:nn {fontspec} { #1 = {##1} } } }
379 }
380 {
381 \keys_if_exist:nnTF {fontspec-preparse} {#1} {
383 \keys_define:nn {fontspec-preparse} { #2 .code:n = { \keys_set:nn {fontspec-preparse} { #1 = {##1} } } }
385 }
386 {
387 \keys_if_exist:nnTF {fontspec-preparse-external} {#1} {
389 \keys_define:nn {fontspec-preparse-external} { #2 .code:n = { \keys_set:nn {fontspec-preparse-external} { #1 = {##1} } } }
391 }
393 {
394 \keys_set:nn {fontspec-preparse-external} { #1 = {##1} } }
396 }
397 \fontspec_warning:nx {rename-feature-not-exist} {#1} }
398 }
399 }
401 }
402 \DeclareDocumentCommand \aliasfontfeatureoption {mmm} {
403 \cs_set_eq:cc { \c_keys_code_root_tl fontspec/#1/#3 } { \c_keys_code_root_tl fontspec/#1/#2 }
406 }

\newfontscript

Mostly used internally, but also possibly useful for users, to define new OpenType ‘scripts’, mapping logical names to OpenType script tags. Iterates through the scripts
in the selected font to check that it’s a valid feature choice, and then prepends the \( \text{XeLaTeX} \) \font feature string with the appropriate script selection tag.

\verb|\DeclareDocumentCommand \newfontscript {mm}|
\verb|{ #1 } { #2 }|
\verb|\fontspec_new_script:nn { #1 } { #2 }|
\verb|\keys_define:nn { fontspec } { Script .choice: }|
\verb|\cs_new:Nn \fontspec_new_script:nn|
\verb|\keys_define:nn { fontspec } { Script / #1 .code:n =|
\verb|\fontspec_check_script:nTF { #2 } |
\verb|\fontspec_update_fontid:n { +script=#1 }|
\verb|\tl_set:Nn \l_fontspec_script_tl { #2 }|
\verbatimcode{\int_set:Nn \l_fontspec_script_int { \l_fontspec_strnum_int }|
\verb|}\}
\verb|\fontspec_check_script:nTF { latin } |
\verbatimcode{\fontspec_warning:nx { script-not-exist } { #1 }|
\verb|\keys_set:nn { fontspec } { Script = Latin }|
\verbatimcode{\}
\verb|\fontspec_warning:nx { script-not-exist } { #1 }|
\verbatimcode{\}}
\verbatimcode{}}

\verb|\newfontlanguage|
Mostly used internally, but also possibly useful for users, to define new OpenType ‘languages’, mapping logical names to OpenType language tags. Iterates through the languages in the selected font to check that it’s a valid feature choice, and then prepends the \( \text{XeLaTeX} \) \font feature string with the appropriate language selection tag.

\verb|\DeclareDocumentCommand \newfontlanguage {mm}|
\verb|{ #1 } { #2 }|
\verbatimcode{\fontspec_new_lang:nn { #1 } { #2 }|
\verbatimcode{\keys_define:nn { fontspec } { Language .choice: }|
\verbatimcode{\keys_define:nn { fontspec } { Language / #1 .code:n =|
\verbatimcode{\fontspec_check_lang:nTF { #2 } |
\verbatimcode{\fontspec_update_fontid:n { +lang=#1 }|
\verbatimcode{\tl_set:Nn \l_fontspec_lang_tl { #2 }|
\verbatimcode{\int_set:Nn \l_fontspec_language_int { \l_fontspec_strnum_int }|
\verbatimcode{\}
\verbatimcode{\fontspec_warning:nx { language-not-exist } { #1 }|
\verbatimcode{\keys_set:nn { fontspec } { Language = Default }|
\verbatimcode{}}
\verbatimcode{}}
\verbatimcode{}}
\verbatimcode{}}

\verbatimcode}
\DeclareFontsExtensions dfont would never be uppercase, right?

\DeclareDocumentCommand \DeclareFontsExtensions {m}
\tl_set:Nn \l_fontspec_extensions_clist { #1 }
\tl_remove_all:Nn \l_fontspec_extensions_clist {˜}
\DeclareFontsExtensions{.otf,.ttf,.OTF,.TTF,.ttc,.TTC,.dfont}

25.3 Programmer’s interface

These functions are not used directly by fontspec when defining fonts; they are designed to be used by other packages who wish to do font-related things on top of fontspec itself.

Because I haven’t fully explored how these functions will behave in practise, I am not giving them user-level names. As it becomes more clear which of these should be accessible by document writers, I’ll open them up a little more.

All functions are defined assuming that the font to be queried is currently selected as a fontspec font. (I.e., via \fontspec or from a \newfontfamily macro or from \setmainfont and so on.)

\fontspec_if_fontspec_font:TF Test whether the currently selected font has been loaded by fontspec.
\fontspec_if_aat_feature:nnTF Conditional to test if the currently selected font contains the AAT feature (#1,#2).
\fontspec_if_opentype:TF Test whether the currently selected font is an OpenType font. Always true for LuaTeX fonts.
\fontspec_if_feature:nTF  
Test whether the currently selected font contains the raw OpenType feature \#1. E.g.: \fontspec_if_feature:nTF {pnum} {True} {False} Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_feature:nnnTF  
Test whether the currently selected font with raw OpenType script tag \#1 and raw OpenType language tag \#2 contains the raw OpenType feature tag \#3. E.g.: \fontspec_if_feature:nnnTF {latn} {ROM} {pnum} {True} {False} Returns false if the font is not loaded by fontspec or is not an OpenType font.

\fontspec_if_script:nTF  
Test whether the currently selected font contains the raw OpenType script \#1. E.g.: \fontspec_if_script:nTF {latn} {True} {False} Returns false if the font is not loaded by fontspec or is not an OpenType font.
\fontspec_if_fontspec_font:TF { \
  \fontspec_font_set:Nnn \l_fontspec_font {\csname zf@family@fontdef\f@family@endcsname} {\f@size pt} \
  \fontspec_set_font_type: \
  \bool_if:NTF \l_fontspec_icu_bool { \
    \fontspec_check_script:nTF {#1} \prg_return_true: \prg_return_false: \
  }{ 
    \prg_return_false: 
  } 
}{ 
  \prg_return_false: 
}

\fontspec_if_language:nTF Test whether the currently selected font contains the raw OpenType language tag #1. E.g.: \fontspec_if_language:nTF {ROM} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.
\prg_new_conditional:Nnn \fontspec_if_language:n {TF,T,F} { 
  \fontspec_if_fontspec_font:TF { 
    \fontspec_font_set:Nnn \l_fontspec_font {\csname zf@family@fontdef\f@family@endcsname} {\f@size pt} 
    \fontspec_set_font_type: 
    \bool_if:NTF \l_fontspec_icu_bool { 
      \tl_set:Nv \l_fontspec_script_tl {g_fontspec_script_(\f@family)_tl} 
      \int_set:Nv \l_fontspec_script_int {g_fontspec_script_num_(\f@family)_tl} 
      \fontspec_check_lang:nTF {#1} \prg_return_true: \prg_return_false: 
    }{ 
      \prg_return_false: 
    } 
  }{ 
    \prg_return_false: 
  }
}

\fontspec_if_language:nnTF Test whether the currently selected font contains the raw OpenType language tag #2 in script #1. E.g.: \fontspec_if_language:nnTF {cyr1} {SRB} {True} {False}. Returns false if the font is not loaded by fontspec or is not an OpenType font.
\prg_new_conditional:Nnn \fontspec_if_language:nn {TF,T,F} { 
  \fontspec_if_fontspec_font:TF { 
    \fontspec_font_set:Nnn \l_fontspec_font {\csname zf@family@fontdef\f@family@endcsname} {\f@size pt} 
    \fontspec_set_font_type: 
    \bool_if:NTF \l_fontspec_icu_bool { 
      \tl_set:Nv \l_fontspec_script_tl {g_fontspec_script_(\f@family)_tl} 
      \fontspec_iv_str_to_num:Nn \l_fontspec_script_int {g_fontspec_script_num_(\f@family)_tl} 
      \fontspec_check_lang:nTF {#1} \prg_return_true: \prg_return_false: 
    }{ 
      \prg_return_false: 
    } 
  }{ 
    \prg_return_false: 
  }
}
Test whether the currently loaded font is using the specified raw OpenType script tag \#1.

559 \prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F} {560 \fontspec_if_fontspec_font:TF {561 \fontspec_font_set:Nnn \l_fontspec_font {\csname zf@family@fontdef\f@family\endcsname} {\f@size pt}562 \fontspec_set_font_type:563 \bool_if:NTF \l_fontspec_icu_bool {564 \str_if_eq:nvTF {#1} {g_fontspec_script_(\f@family)_tl}565 \prg_return_true:} \prg_return_false:566 \prg_return_false:567 }568 }569 \prg_return_false:570 \prg_return_false:571 )572 }

Test whether the currently loaded font is using the specified raw OpenType language tag \#1.

573 \prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F} {574 \fontspec_if_fontspec_font:TF {575 \fontspec_font_set:Nnn \l_fontspec_font {\csname zf@family@fontdef\f@family\endcsname} {\f@size pt}576 \fontspec_set_font_type:577 \bool_if:NTF \l_fontspec_icu_bool {578 \str_if_eq:nvTF {#1} {g_fontspec_lang_(\f@family)_tl}579 \prg_return_true:} \prg_return_false:580 \prg_return_false:581 \prg_return_false:582 )583 \prg_return_false:584 \prg_return_false:585 )586 }

\fontspec_set_family:Nnn \#1 : family \#2 : fontspec features \#3 : font name

Defines a new font family from given \langle features \rangle and \langle font \rangle, and stores the name in the variable \langle family \rangle. See the standard fontspec user commands for applications of this function.

We want to store the actual name of the font family within the \langle family \rangle variable because the actual \LaTeX family name is automatically generated by fontspec and it’s easier to keep it that way.

Please use \fontspec_set_family:Nnn instead of \fontspec_select:nn, which may change in the future.

587 \cs_new:Nn \fontspec_set_family:Nnn {588 \fontspec_select:nn (#2)(#3)589 \tl_set_eq:NN \#1 \l_fontspec_family_tl590 )
25.4 expl3 interface for font loading

Beginnings of an 'l3font', I guess:
\begin{verbatim}
\cs_set:Nn \font_set_eq:NN { #1#2#3 { \font #1 = #2 \space at \space #3 \scan_stop: }
\cs_set:Npn \font_set:Nnn #1#2#3 { \global \font #1 = #2 \space at \space #3 \scan_stop: }
\cs_set:Npn \font_suppress_not_found_error: { \langle xetex \rangle { \suppressfontnotfounderror=1 } \langle luatex \rangle { \luatexsuppressfontnotfounderror=1 } \prg_set_conditional:Nnn \font_if_null:N { p,TF,T,F } { \ifx #1 \nullfont \prg_return_true: \else \prg_return_false: \fi } }
\end{verbatim}

Wrapper around \font_set:Nnn and \font_gset:Nnn.
\begin{verbatim}
\cs_new:Nn \fontspec_font_set:Nnn { \font_set:Nnn #1 { \fontspec_fontwrap:n {#2}} {#3} }
\cs_new:Nn \fontspec_font_gset:Nnn { \font_gset:Nnn #1 { \fontspec_fontwrap:n {#2}} {#3} }
\end{verbatim}

\begin{verbatim}
\fontGlyph_if_exist:NnTF
\prg_new_conditional:Nnn \font Glyph_if_exist:Nn { p,TF,T,F } { \etex_iffontchar:D #1 #2 \scan_stop: \prg_return_true: \else: \prg_return_false: \fi: }
\end{verbatim}
25.5 Internal macros

The macros from here in are used internally by all those defined above. They are not designed to remain consistent between versions.

\fontspec_select:nn

This is the command that defines font families for use, the underlying procedure of all \fontspec-like commands. Given a list of font features (#1) for a requested font (#2), it will define an NFSS family for that font and put the family name (globally) into $\l_{\text{fontspec\_family\_tl}}$. The \TeX \textbackslash font command is (globally) stored in $\l_{\text{fontspec\_font}}$.

This macro does its processing inside a group to attempt to restrict the scope of its internal processing. This works to some degree to insulate the internal commands from having to be manually cleared.

\cs_set:Nn \fontspec_select:nn {
\group_begin:
\font_suppress_not_found_error:
\fontspec_init:

$\l_{\text{fontspec\_fontname\_tl}}$ is used as the generic name of the font being defined. $\l_{\text{fontspec\_fontid\_tl}}$ is the unique identifier of the font with all its features. $\l_{\text{fontspec\_fontname\_up\_tl}}$ is the font specifically to be used as the upright font.

\tl_set:Nx \l_fontspec_fontname_tl {#2}
⟨luatex⟩ \tl_remove_all:Nn \l_fontspec_fontname_tl {˜}
\tl_set_eq:NN \l_fontspec_fontid_tl \l_fontspec_fontname_tl
\tl_set_eq:NN \l_fontspec_fontname_up_tl \l_fontspec_fontname_tl

Now convert the requested features to font definition strings. First the features are parsed for information about font loading (whether it’s a named font or external font, etc.), and then information is extracted for the names of the other shape fonts.

Then the mapping from user features to low-level features occurs. This is performed with \fontspec_get_features:n, in which \keys_set:nn retrieves the requested font features and processes them. As \keys_set:nn is run multiple times, some of its information storing only occurs once while we decide if the font family has been defined or not. When the later processing is occurring per-shape this no longer needs to happen; this is indicated by the ‘firsttime’ conditional.

\exp_args:NnV \fontspec_preparse_features:nn {#1} \l_fontspec_fontname_tl

Finally save the ‘confirmed’ font definition.

\fontspec_font_set:Nnn \l_fontspec_font {\fontspec_fullname:n \l_fontspec_fontname_up_tl} \f@size pt
\font_if_null:NT \l_fontspec_font {\fontspec_error:nx {font-not-found} \l_fontspec_fontname_up_tl}
\fontspec_set_font_type:
\fontspec_font_gset:Nnn \l_fontspec_font {\fontspec_fullname:n \l_fontspec_fontname_up_tl}
\l_fontspec_font % this is necessary for LuaLaTeX to check the scripts properly

Continue:

\fontspec_set_scriptlang:
\fontspec_get_features:n {}\bool_set_false:N \l_fontspec_firsttime_bool

Check if the family is unique and, if so, save its information. (\addfontfeature and other macros use this data.) Then the font family and its shapes are defined in the NFSS.
All NFSS specifications take their default values, so if any of them are redefined, the shapes will be selected to fit in with the current state. For example, if \textbf{default} is redefined to \textit{b}, all bold shapes defined by this package will also be assigned to \textit{b}.

\begin{verbatim}
647 \fontspec_save_family:nT (#2) { 
648 \fontspec_save_fontinfo:nn {#1} (#2) 
649 \DeclareFontFamily{\g_fontspec_encoding_tl}{\l_fontspec_family_tl}{ }
650 \fontspec_set_upright: 
651 \fontspec_set_bold: 
652 \fontspec_set_italic: 
653 \fontspec_set_slanted: 
654 \fontspec_set_bold_italic: 
655 \fontspec_set_bold_slanted: 
656 } 
657 \fontspec_info:nxx {defining-font} {#1} {#2} 
658 \group_end: 
659 }
\end{verbatim}

\fontspec_preparse_features:nn Perform the (multi-step) feature parsing process.

\begin{verbatim}
660 \cs_new:Nn \fontspec_preparse_features:nn { 
661 \fontspec_if_detect_external:nT {#2} { 
662 \keys_set:nn {fontspec-preparse-external} {ExternalLocation} 
663 \keys_set_known:nxN {fontspec-preparse-external} {\g_fontspec_default_fontopts_tl #1} \l_fontspec_keys_leftover_clist 
664 \tl_set_eq:NN \l_fontspec_fontname_tl \l_fontspec_fontname_up_tl 
665 \keys_set_known:nxN {fontspec-preparse} {\l_fontspec_keys_leftover_clist} 
666 \l_fontspec_fontfeat_clist 
667 
668 }
\end{verbatim}

\fontspec_if_detect_external:nT Check if either the fontname ends with a known font extension.

\begin{verbatim}
669 \prg_new_conditional:Nnn \fontspec_if_detect_external:n {T} 
670 { 
671 \clist_map_inline:Nn \l_fontspec_extensions_clist 
672 { 
673 \bool_set_false:N \l_tmpa_bool 
674 \tl_if_in:nNnT {#1 <= end_of_string} {##1 <= end_of_string} 
675 { \bool_set_true:N \l_tmpa_bool \clist_map_break: } 
676 } 
677 \bool_if:NTF \l_tmpa_bool \prg_return_true: \prg_return_false: 
678 }
\end{verbatim}

\fontspec_fullname:n Constructs the complete font name based on a common piece of info.

\begin{verbatim}
679 \cs_set:Nn \fontspec_fullname:n { 
680 \fontspec_namewrap:n { #1 \l_fontspec_extension_tl } 
681 \l_fontspec_renderer_tl 
682 }
\end{verbatim}
Now we have a unique (in fact, too unique!) string that contains the family name and every option in abbreviated form. This is used with a counter to create a simple NFSS family name for the font we’re selecting.

The font name is fully expanded, in case it’s defined in terms of macros, before having its spaces zapped.

\fontspec_set_scriptlang: Only necessary for OpenType fonts. First check if the font supports scripts, then apply defaults if none are explicitly requested. Similarly with the language settings.

\fontspec_save_family:nT
\keys_set:nx {fontspec} {Language=\l_fontspec_lang_name_tl}
}
}
}
\fontspec_save_fontinfo:nn
Saves the relevant font information for future processing.
\cs_generate_variant:Nn \prop_gput:Nnn {cnV}
\cs_generate_variant:Nn \prop_gput:Nnn {cnx}
\cs_new:Nn \fontspec_save_fontinfo:nn {
\prop_new:c {g_fontspec_\l_fontspec_family_tl_prop}
\prop_gput:cnx {g_fontspec_\l_fontspec_family_tl_prop} {fontname} {#2}
\prop_gput:cnx {g_fontspec_\l_fontspec_family_tl_prop} {options} \g_fontspec_default_fontopts_tl #1
\prop_gput:cnx {g_fontspec_\l_fontspec_family_tl_prop} {fontdef} {
\fontspec_fullname:n {\l_fontspec_fontname_tl} :
\l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist}
\prop_gput:cnV {g_fontspec_\l_fontspec_family_tl_prop} {script-num} \l_fontspec_script_int
\prop_gput:cnV {g_fontspec_\l_fontspec_family_tl_prop} {lang-num} \l_fontspec_language_int
\prop_gput:cnV {g_fontspec_\l_fontspec_family_tl_prop} {script-tag} \l_fontspec_script_tl
\prop_gput:cnV {g_fontspec_\l_fontspec_family_tl_prop} {lang-tag} \l_fontspec_lang_tl
}
\tl_gset:cx {zf@family@fontname\l_fontspec_family_tl} {#2}
\tl_gset:cx {zf@family@options\l_fontspec_family_tl} \g_fontspec_default_fontopts_tl #1
\tl_gset:cx {zf@family@fontdef\l_fontspec_family_tl} {
\fontspec_fullname:n {\l_fontspec_fontname_tl} :
\l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist}
\tl_gset:cx {zf@family@fontname\l_fontspec_family_tl} (#2)
\tl_gset:cx {zf@family@options\l_fontspec_family_tl} \g_fontspec_default_fontopts_tl #1
\tl_gset:cx {zf@family@fontdef\l_fontspec_family_tl} {
\fontspec_fullname:n {\l_fontspec_fontname_tl} :
\l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist}
\tl_gset:cx {zf@family@fontname\l_fontspec_family_tl} (#2)
\tl_gset:cx {zf@family@options\l_fontspec_family_tl} \g_fontspec_default_fontopts_tl #1
\tl_gset:cx {zf@family@fontdef\l_fontspec_family_tl} {
\fontspec_fullname:n {\l_fontspec_fontname_tl} :
\l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist}
\fontspec_set_upright: Sets the upright shape.
\cs_new:Nn \fontspec_set_upright: {
\fontspec_make_font_shapes:nnnn \l_fontspec_fontname_tl
\mddefault \updefault \l_fontspec_fontfeat_up_clist
}
\fontspec_set_bold: The macros [...] et al., are used to store the name of the custom bold, et al., font, if requested as user options. If they are empty, the default fonts are used.
The extra bold options defined with BoldFeatures are appended to the generic font features. Then, the bold font is defined either as the ATS default (function argument is to check if there actually is one; if not, the bold NFSS series is left undefined) or with the font specified with the BoldFont feature.
\cs_new:Nn \fontspec_set_bold: {
\bool_if:NF \l_fontspec_nobf_bool {
\tl_if_empty:NTF \l_fontspec_fontname_bf_tl {
\fontspec_make_auto_font_shapes:nnnn \l_fontspec_fontname_tl (/B)
\mddefault \updefault \l_fontspec_fontfeat_bf_clist
}}
\fontspec_set_italic:  And italic in the same way:
\cs_new:Nn \fontspec_set_italic: { 
\bool_if:NF \l_fontspec_noit_bool { 
\tl_if_empty:NTF \l_fontspec_fontname_it_tl { \fontspec_make_auto_font_shapes:nnnn \l_fontspec_fontname_tl {/I} } 
{ \fontspec_make_font_shapes:nnnn \l_fontspec_fontname_tl } \mddefault \itdefault \l_fontspec_fontfeat_it_clist 
} 
}
\fontspec_set_slanted:  And slanted but only if requested:
\cs_new:Nn \fontspec_set_slanted: { 
\tl_if_empty:NF \l_fontspec_fontname_sl_tl { 
\fontspec_make_font_shapes:nnnn \l_fontspec_fontname_tl \mddefault \sldefault \l_fontspec_fontfeat_sl_clist 
} 
}
\fontspec_set_bold_italic:  If requested, the custom fonts take precedence when choosing the bold italic font.
When both italic and bold fonts are requested and the bold italic font hasn't been explicitly specified (a rare occurrence, presumably), the new bold font is used to define the new bold italic font.
\cs_new:Nn \fontspec_set_bold_italic: { 
\bool_if:nF { \l_fontspec_noit_bool || \l_fontspec_nobf_bool } { 
\tl_if_empty:NTF \l_fontspec_fontname_bfit_tl { 
\tl_if_empty:NTF \l_fontspec_fontname_bf_tl { 
\tl_if_empty:NTF \l_fontspec_fontname_it_tl { 
\fontspec_make_auto_font_shapes:nnnnn \l_fontspec_fontname_tl {/BI} } 
} 
\fontspec_make_auto_font_shapes:nnnn \l_fontspec_fontname_tl {/B} } 
} 
}
And bold slanted, again, only if requested:
\fontspec_set_bold_slanted:
\cs_new:Nn \fontspec_set_bold_slanted:
\tl_if_empty:NTF \l_fontspec_fontname_bfsl_tl
\tl_if_empty:NF \l_fontspec_fontname_sl_tl {
\fontspec_make_auto_font_shapes:nnnn \l_fontspec_fontname_sl_tl (/B)
\bfdefault \sldefault \l_fontspec_fontfeat_bfsl_clist
}\}
\fontspec_make_font_shapes:nnnn \l_fontspec_fontname_bfsl_tl
\bfdefault \sldefault \l_fontspec_fontfeat_bfsl_clist
}\}

\fontspec_set_font_type:  Now check if the font is to be rendered with \atsui or \icu. This will either be automatic (based on the font type), or specified by the user via a font feature.
This macro sets booleans accordingly depending if the font in \l_fontspec_font is an \aat font or an OpenType font or a font with feature axes (either \aat or Multiple Master), respectively.
\fontspec_set_font_type:
\cs_new:Nn \fontspec_set_font_type:
\begin{c旦tex}
\bool_set_false:N \l_fontspec_tfm_bool
\bool_set_false:N \l_fontspec_atsui_bool
\bool_set_false:N \l_fontspec_icu_bool
\bool_set_false:N \l_fontspec_mm_bool
\bool_set_false:N \l_fontspec_graphite_bool
\ifcase\XeTeXfonttype\l_fontspec_font
\bool_set_true:N \l_fontspec_tfm_bool
\or
\bool_set_true:N \l_fontspec_atsui_bool
\ifnum\XeTeXcountvariations\l_fontspec_font > \c_zero
\bool_set_true:N \l_fontspec_mm_bool
\fi
\or
\bool_set_true:N \l_fontspec_icu_bool
\fi

If automatic, the \l_fontspec_renderer_tl token list will still be empty (other suffices that could be added will be later in the feature processing), and if it is indeed still empty, assign it a value so that the other weights of the font are specifically loaded with the same renderer.
This macro eventually uses \DeclareFontShape to define the font shape in question.

The optional first argument is used when making the font shapes for bold, italic, and bold italic fonts using Xe\TeX’s auto-recognition with #2 as /B, /I, and /BI font name suffixes. If no such font is found, it falls back to the original font name, in which case this macro doesn’t proceed and the font shape is not created for the NFSS.

Next, the small caps are defined. […] is used to define the appropriate string for activating small caps in the font, if they exist. If we are defining small caps for the upright shape, then the small caps shape default is used. For an italic font, however, the shape parameter is overloaded and we must call italic small caps by their own identifier. See Section 25.7 on page 108 for the code that enables this usage.
Note that the test for italics to choose the \sidefault shape only works while \fontspec_select:nn passes single tokens to this macro...

\fontspec_declare_shape:nnn

#1 : Raw appended font features
#2 : Font series
#3 : Font shape
#4 : Font features

Wrapper for \DeclareFontShape.

\cs_new:Nn \fontspec_declare_shape:nnn { ... }
\clist_if_empty:NTF \l_fontspec_sizefeat_clist { ... }
\tl_clear:N \l_fontspec_nfss_tl
\clist_map_inline:Nn \l_fontspec_sizefeat_clist { ... }
\tl_clear:N \l_fontspec_size_tl
\tl_set_eq:NN \l_fontspec_sizedfont_tl \l_fontspec_fontname_tl
\fontspec_fontwrap:n { ... }
\fontspec_fullname:n { \l_fontspec_fontname_tl : \l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist }
\fontspec_font:nnn { ... }
\tl_clear:N \l_fontspec_nfss_tl
\clist_map_inline:Nn \l_fontspec_sizefeat_clist { ... }
\tl_set_eq:NN \l_fontspec_sizedfont_tl \l_fontspec_fontname_tl
\fontspec_fontwrap:n { ... }
\fontspec_fullname:n { \l_fontspec_fontname_tl : \l_fontspec_pre_feat_sclist \l_fontspec_rawfeatures_sclist }
\fontspec_font:nnn { ... }
\group_end:...
And finally the actual font shape declaration using \_\_fontspec\_nfss\_tl defined above. \_\_fontspec\_postadjust\_tl is defined in various places to deal with things like the hyphenation character and interword spacing.

\use:x{
\exp_not:N\DeclareFontShape{\g_fontspec_encoding_tl}{\l_fontspec_family_tl}{#1}{#2}{\l_fontspec_nfss_tl}{\l_fontspec_postadjust_tl}
}

This extra stuff for the slanted shape substitution is a little bit awkward, but I’d rather have it here than break out yet another macro.

\bool_if:nT { \str_if_eq_p:xx {#2} {\itdefault} && !(\str_if_eq_p:xx {\itdefault} {\sldefault}) }
\use:x {
\exp_not:N\DeclareFontShape{\g_fontspec_encoding_tl}{\l_fontspec_family_tl}{#1}{\sldefault}{<->ssub*\l_fontspec_family_tl/#1/\itdefault}{\l_fontspec_postadjust_tl}
}

Lastly some informative messaging.

\tl_gput_right:Nx \_\_\_fontspec\_defined\_shapes\_tl
{ \exp_not:n { \ \ \ \ }
\"\"\exp_not:N \prg_case_str:nnn {#1/#2} {#1/#2} {\\}
{ \mddefault/\updefault} {normal}
{ \mddefault/\scdefault} {small~ caps}
{ \bfdefault/\updefault} {bold}
{ \bfdefault/\scdefault} {bold~ small~ caps}
{ \mddefault/\itdefault} {italic}
{ \mddefault/\sidefault} {italic~ small~ caps}
{ \bfdefault/\itdefault} {bold~ italic}
{ \bfdefault/\sidefault} {bold~ italic~ small~ caps}
} \{#2/#3\}

with~ NFSS~ spec.: \exp_not:N \"
These are the features always applied to a font selection before other features.

This macro is used to build up a complex family name based on its features.

The \firsttime boolean is set true in \fontspec_select:nn only the first time \fontspec_update_fontid:n is called, so that the family name is only created once.

This macro is a wrapper for \keys_set:nn which expands and adds a default specification to the original passed options. It begins by initialising the commands used to hold font-feature specific strings. Its argument is any additional features to prepend to the default.
\tl_set_eq:NN \l_fontspec_hexcol_tl \g_fontspec_hexcol_tl
\tl_clear:N \l_fontspec_postadjust_tl
\keys_set:nx {fontspec} {\l_fontspec_fontfeat_clist, #1}

Finish the colour specification. Do not set the colour if not explicitly spec’d else \color (using specials) will not work.
\str_if_eq:xxF { \l_fontspec_hexcol_tl \l_fontspec_opacity_tl } { \g_fontspec_hexcol_tl \g_fontspec_opacity_tl } {
\fontspec_update_featstr:n{color=\l_fontspec_hexcol_tl\l_fontspec_opacity_tl}
}
\fontspec_init:
Initialisations that either need to occur globally: (all setting of these variables is done locally inside a group)
\tl_clear:N \l_fontspec_fontname_bf_tl
\tl_clear:N \l_fontspec_fontname_it_tl
\tl_clear:N \l_fontspec_fake_slant_tl
\tl_clear:N \l_fontspec_fake_embolden_tl
\tl_clear:N \l_fontspec_fontname_bfit_tl
\tl_clear:N \l_fontspec_fontname_sl_tl
\tl_clear:N \l_fontspec_fontname_bfsl_tl
\tl_clear:N \l_fontspec_fontname_sc_tl
\tl_clear:N \l_fontspec_fontfeat_up_clist
\tl_clear:N \l_fontspec_fontfeat_bf_clist
\tl_clear:N \l_fontspec_fontfeat_it_clist
\tl_clear:N \l_fontspec_fontfeat_bfit_clist
\tl_clear:N \l_fontspec_fontfeat_sl_clist
\tl_clear:N \l_fontspec_fontfeat_bfsl_clist
\tl_clear:N \l_fontspec_fontfeat_sc_clist
\tl_clear:N \l_fontspec_script_name_tl
\tl_clear:N \l_fontspec_script_tl
\clist_clear:N \l_fontspec_sizefeat_clist
\tl_new:N \g_fontspec_hexcol_tl
\tl_new:N \g_fontspec_opacity_tl
\tl_set:Nn \g_fontspec_hexcol_tl {/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash}
\tl_set:Nn \g_fontspec_opacity_tl {FF˜}

Or once per fontspec font invocation: (Some of these may be redundant. Check whether they’re assigned to globally or not.)
\cs_set:Npn \fontspec_init: {
\bool_set_false:N \l_fontspec_icu_bool
\bool_set_true:N \l_fontspec_firsttime_bool
\cs_set:Npn \fontspec_namewrap:n ##1
⟨\xetexx⟩ { ##1 }
⟨\luatex⟩ { name:##1 }
\tl_clear:N \l_fontspec_optical_size_tl
\tl_clear:N \l_fontspec_renderer_tl
\tl_clear:N \l_fontspec_defined_shapes_tl
⟨∗\luatex⟩ 79

\l_set:Nn \l_fontspec_mode_tl {node}
\luatexprehyphenchar ='|- % fixme
\luatexposthyphenchar = 0 % fixme
\luatexpreexhyphenchar = 0 % fixme
\luatexpostexhyphenchar= 0 % fixme
\fontspec_make_smallcaps:T This macro checks if the font contains small caps.
\cs_set:Nn \fontspec_make_ot_smallcaps:T {
  \fontspec_check_ot_feat:nT {+smcp} { #1 }
}
\cs_set:Nn \fontspec_make_smallcaps:T {
  \bool_if:NTF \l_fontspec_icu_bool {
    \fontspec_make_ot_smallcaps:T {#1}
  }{
    \bool_if:NT \l_fontspec_atsui_bool {
      \fontspec_make_AAT_feature_string:nnT {3}{3} { #1 }
    }
  }
}
\cs_set_eq:NN \fontspec_make_smallcaps:T \fontspec_make_ot_smallcaps:T
\fontspec_update_featstr:n \l_fontspec_rawfeatures_sclist is the string used to define the list of specific font features. Each time another font feature is requested, this macro is used to add that feature to the list. Font features are separated by semicolons.
\cs_set_eq:NN \sclist_clear:N \tl_clear:N
\cs_new:Nn \sclist_gput_right:Nn { \tl_gput_right:Nn {#1}; }
\cs_generate_variant:Nn \sclist_gput_right:Nn {Nx}
\fontspec_make_feature:nnn This macro is called by each feature key selected, and runs according to which type of font is selected.
\cs_set_eq:NN \sclist_clear:N \tl_clear:N
\cs_new:Nn \sclist_gput_right:Nn { \tl_gput_right:Nn {#1}; }
\cs_generate_variant:Nn \sclist_gput_right:Nn {Nx}
\sclist_put_right:Nn I'm hardly going to write an 'sclist' module but a couple of functions are useful. Here, items in semi-colon lists are always followed by a semi-colon (as opposed to the s.-c's being placed between elements) so we can append sclists without worrying about it.
\cs_set_eq:NN \sclist_clear:N \tl_clear:N
\cs_new:Nn \sclist_gput_right:Nn { \tl_gput_right:Nn {#1}; }
\cs_generate_variant:Nn \sclist_gput_right:Nn {Nx}
\bool_if:NT \l_fontspec_icu_bool {
  \fontspec_make_ICU_feature:n (#3)
}\}
\bool_if:NT \l_fontspec_atsui_bool {
  \fontspec_make_AAT_feature:nn (#1)(#2)
}\}
\fontspec_make_ICU_feature:n (#3)
\langle \texttt{xetexx} \rangle
\langle \texttt{luatex} \rangle
\fontspec_make_ICU_feature:n (#3)
\langle \texttt{xetexx} \rangle
\langle \texttt{luatex} \rangle
\cs_generate_variant:Nn \fontspec_make_feature:nnn {nnx}
\cs_new:Nn \fontspec_make_AAT_feature:nn {
  \tl_if_empty:nTF {#1}
  { \fontspec_warning:n {aat-feature-not-exist} }
  {
    \fontspec_make_AAT_feature_string:nnTF {#1}{#2}
    { \fontspec_update_fontid:n {+#1,#2}
      \fontspec_update_featstr:n {\l_fontspec_feature_string_tl}
    }
    { \fontspec_warning:nx {aat-feature-not-exist-in-font} {#1,#2} }
  }
}\}
\cs_new:Nn \fontspec_make_ICU_feature:n {
  \tl_if_empty:nTF {#1}
  { \fontspec_warning:n {icu-feature-not-exist} }
  {
    \fontspec_check_ot_feat:nTF {#1}
    { \fontspec_update_fontid:n {#1}
      \fontspec_update_featstr:n {\l_fontspec_feature_string_tl}
    }
    { \fontspec_warning:nx {icu-feature-not-exist-in-font} {#1} }
  }
}\}
\cs_new_protected:Nn \fontspec_make_numbered_feature:nn {
  \fontspec_check_ot_feat:nTF {#1}
  { \fontspec_update_fontid:n {#1=#2}
    \langle \texttt{xetexx} \rangle \fontspec_update_featstr:n { #1 = #2 }
    \langle \texttt{luatex} \rangle \fontspec_update_featstr:n { #1 = \int_eval:n {#2+1} }
  }
  { \fontspec_warning:nx {icu-feature-not-exist-in-font} {#1} }
}\}
\cs_generate_variant:Nn \fontspec_make_numbered_feature:nn {xn}
These macros are used in order to simplify font feature definition later on.

\cs_new:Nn \fontspec_define_font_feature:n { \keys_define:nn {fontspec} { #1 .multichoice: } }
\cs_new:Nn \fontspec_define_feature_option:nnnnn { \keys_define:nn {fontspec} { #1/#2 .code:n = \fontspec_make_feature:nnn{#3}{#4}{#5} } }
\cs_new:Nn \fontspec_define_numbered_feat:nnnn { \keys_define:nn {fontspec} { #1/#2 .code:n = \fontspec_make_numbered_feature:nn {#3}{#4} } }

This macro takes the numerical codes for a font feature and creates a specified macro containing the string required in the font definition to turn that feature on or off. Used primarily in [...] but also used to check if small caps exists in the requested font (see page 80).

For exclusive selectors, it's easy; just grab the string: For non-exclusive selectors, it's a little more complex. If the selector is even, it corresponds to switching the feature on. If the selector is odd, it corresponds to switching the feature off. But Xe\TeX{} doesn't return a selector string for this number, since the feature is defined for the ‘switching on’ value. So we need to check the selector of the previous number, and then prefix the feature string with \texttt{!} to denote the switch.

Finally, save out the complete feature string in \l_fontspec_feature_string_tl.
This macro takes a four character string and converts it to the numerical representation required for XeLaTeX OpenType script/language/feature purposes. The output is stored in \l_fontspec_strnum_int.

The reason it’s ugly is because the input can be of the form of any of these: ‘abcd’, ‘abc’, ‘abc’, ‘ab’, ‘ab’, etc. (It is assumed the first two chars are always not spaces.) So this macro reads in the string, delimited by a space; this input is padded with \@empty and anything beyond four chars is snipped. The \@empty s then are used to reconstruct the spaces in the string to number calculation.

The variant \fontspec_v_str_to_num:n is used when looking at features, which are passed around with prepended plus and minus signs (e.g., +liga, -dlig); it simply strips off the first char of the input before calling the normal \fontspec_iv_str_to_num:n.

\fontspec_check_script:nTF

This macro takes an OpenType script tag and checks if it exists in the current font. The output boolean is \@tempswatrue. \l_fontspec_strnum_int is used to store the number corresponding to the script tag string.
This macro takes an OpenType language tag and checks if it exists in the current font/script. The output boolean is `\@tempswatrue`. `\l_fontspec_strnum_int` is used to store the number corresponding to the language tag string. The script used is whatever's held in `\l_fontspec_script_int`. By default, that's the number corresponding to 'latn'.

```latex
\fontspec_check_lang:nTF
```

84
This macro takes an OpenType feature tag and checks if it exists in the current font/script/language. The output boolean is \@tempswa. \l_fontspec_strnum_int is used to store the number corresponding to the feature tag string. The script used is whatever’s held in \l_fontspec_script_int. By default, that’s the number corresponding to ‘latn’. The language used is \l_fontspec_language_int, by default /zero.noslash, the ‘default language’.

\fontspec_check_ot_feat:nTF
\fontspec_check_ot_feat:nT

\fontspec_check_ot_feat:nTF
\fontspec_check_ot_feat:nT

This is the tedious section where we correlate all possible (eventually) font feature requests with their \TeX representations.

25.6 keyval definitions

This is the tedious section where we correlate all possible (eventually) font feature requests with their \TeX representations.
25.6.1 Pre-parsing naming information

These features are extracted from the font feature list before all others.

ExternalLocation   For fonts that aren’t installed in the system. If no argument is given, the font is located with kpsewhich; it’s either in the current directory or the \TeX tree. Otherwise, the argument given defines the file path of the font.

\bool_new:N \l_fontspec_external_bool
\keys_define:nn {fontspec-preparse-external} {
  \ExternalLocation .code:n = {
    \bool_set_true:N \l_fontspec_nobf_bool
    \bool_set_true:N \l_fontspec_noit_bool
    \bool_set_true:N \l_fontspec_external_bool
    \cs_gset:Npn \fontspec_namewrap:n ##1
      ⟨xetex⟩{ [ #1 ##1 ] }
      ⟨luatex⟩{ file: #1 ##1 }
      ⟨∗xetex⟩
    \keys_set:nn {fontspec-preparse} {Renderer=ICU}
  }\langle/xetex⟩
\tl_clear:N \l_fontspec_extension_tl
}\aliasfontfeature{ExternalLocation}{Path}

Extension   For fonts that aren’t installed in the system. Specifies the font extension to use.

\keys_define:nn {fontspec-preparse-external} {
  Extension .code:n = {
    \tl_set:Nn \l_fontspec_extension_tl {#1}
    \bool_if:NF \l_fontspec_external_bool {
      \keys_set:nn {fontspec-preparse-external} {ExternalLocation}
    }
  }
}\aliasfontfeature{Extension}{Path}

25.6.2 Pre-parsed features

After the font name(s) have been sorted out, now need to extract any renderer/font configuration features that need to be processed before all other font features.

Renderer   This feature must be processed before all others (the other font shape and features options are also pre-parsed for convenience) because the renderer determines the format of the features and even whether certain features are available.

\keys_define:nn {fontspec-preparse} {
  Renderer .choice_code:n = {
    \fontspec_update_fontid:n {+rend:\l_keys_choice_tl}
    \int_compare:nTF {\l_keys_choice_int < 3} {
      (+xetexx)
    }
  }
}
OpenType script/language  See later for the resolutions from fontspec features to OpenType definitions.

Exactly the same:

25.6.3 Bold/italic choosing options

The Bold, Italic, and BoldItalic features are for defining explicitly the bold and italic fonts used in a font family.

Fonts  Upright:
Bold:
\keys_define:nn {fontspec-preparse-external} { BoldFont .code:n = ( \tl_if_empty:nTF (#1) ( \bool_set_true:N \l_fontspec_nobf_bool \fontspec_update_fontid:n {nobf} ) \bool_set_false:N \l_fontspec_nobf_bool \fontspec_complete_fontname:Nn \l_fontspec_fontname_bf_tl {#1} \fontspec_update_fontid:n {bf:#1} )}
\keys_define:nn {fontspec-preparse-external} { SlantedFont .code:n = ( \fontspec_complete_fontname:Nn \l_fontspec_fontname_sl_tl (#1) \fontspec_update_fontid:n {sl:#1} )}

Same for italic:
\keys_define:nn {fontspec-preparse-external} { ItalicFont .code:n = ( \tl_if_empty:nTF (#1) ( \bool_set_true:N \l_fontspec_noit_bool \fontspec_update_fontid:n {noit} ) \bool_set_false:N \l_fontspec_noit_bool \fontspec_complete_fontname:Nn \l_fontspec_fontname_it_tl {#1} \fontspec_update_fontid:n {it:#1} )}
\keys_define:nn {fontspec-preparse-external} { BoldSlantedFont .code:n = ( \fontspec_complete_fontname:Nn \l_fontspec_fontname_bfsl_tl (#1) \fontspec_update_fontid:n {bfsl:#1} )}

Simpler for bold+italic & slanted:
\keys_define:nn {fontspec-preparse-external} { BoldItalicFont .code:n = ( \fontspec_complete_fontname:Nn \l_fontspec_fontname_bfit_tl (#1) \fontspec_update_fontid:n {bfit:#1} )}
\keys_define:nn {fontspec-preparse-external} { BoldSlantedFont .code:n = ( \fontspec_complete_fontname:Nn \l_fontspec_fontname_bfsl_tl (#1) \fontspec_update_fontid:n {bfsl:#1} )}

Small caps isn’t pre-parsed because it can vary with others above:
This macro defines #1 as the input with any * tokens of its input replaced by the font name. This lets us define supplementary fonts in full (“Baskerville Semibold”) or in abbreviation (“* Semibold”).

Features  Can’t use \clist_set:Nn below, yet, because it would strip the leading comma and we use that implicitly to concatenate options.

\keys_define:nn {fontspec-preparse} { UprightFeatures .code:n = { \tl_set:Nn \l_fontspec_fontfeat_up_clist {, #1} \fontspec_update_fontid:n {rmfeat:#1} } } 
\keys_define:nn {fontspec-preparse} { BoldFeatures .code:n = { \tl_set:Nn \l_fontspec_fontfeat_bf_clist {, #1} \fontspec_update_fontid:n {bffeat:#1} } } 
\keys_define:nn {fontspec-preparse} { ItalicFeatures .code:n = { \tl_set:Nn \l_fontspec_fontfeat_it_clist {, #1} \fontspec_update_fontid:n {itfeat:#1} } } 
\keys_define:nn {fontspec-preparse} { BoldItalicFeatures .code:n = { \tl_set:Nn \l_fontspec_fontfeat_bfit_clist {, #1} \fontspec_update_fontid:n {bfitfeat:#1} } } 
\keys_define:nn {fontspec-preparse} { SlantedFeatures .code:n = { \tl_set:Nn \l_fontspec_fontfeat_sl_clist {, #1} \fontspec_update_fontid:n {slfeat:#1} } } 
\keys_define:nn {fontspec-preparse} { BoldSlantedFeatures .code:n = {
Note that small caps features can vary by shape, so these in fact aren’t pre-parsed.

paragraphFeatures varying by size TODO: sizezfeatures and italicfont (etc) don’t play nice

25.6.4 Font-independent features

These features can be applied to any font.

Scale If the input isn’t one of the pre-defined string options, then it’s gotta be numerical. \fontspec\texttt{calc_scale:n} does all the work in the auto-scaling cases.
This macro calculates the amount of scaling between the default roman font and
the (default shape of) the font being selected such that the font dimension that is
input is equal for both. The only font dimensions that justify this are 5 (lowercase
height) and 8 (uppercase height in \TeX).

This script is executed for every extra shape, which seems wasteful, but allows
alternate italic shapes from a separate font, say, to be loaded and to be auto-scaled
correctly. Even if this would be ugly.

\fontspec_calc_scale:n

This function sets the dimension \#1 (for font \#3) to \fontdimen\#2 for either font
dimension 5 (x-height) or 8 (cap-height). If, for some reason, these return an
incorrect ‘zero’ value (as \fontdimen8 might for a .tfm font), then we cheat and
measure the height of a glyph. We assume in this case that the font contains either
an ‘X’ or an ‘x’.

\fontspec_set_font_dimen:NnN

\keys_define:nn {fontspec} { WordSpace .code:n = {
  \fontspec_update_fontid:n {+wordspace:#1}
  \bool_if:NF \l_fontspec_firsttime_bool {
    _fontspec_parse_wordspace:w #1,,\q_stop
  }
}}
This macro determines if the input to WordSpace is of the form \{X\} or \{X,Y,Z\} and executes the font scaling. If the former input, it executes \{X,X,X\}.

\cs_set:Npn \_fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop {
  \tl_if_empty:nTF {#4} {
    \tl_put_right:Nn \l_fontspec_postadjust_tl {
      \fontdimen 2 \font = #1 \fontdimen 2 \font
      \fontdimen 3 \font = #1 \fontdimen 3 \font
      \fontdimen 4 \font = #1 \fontdimen 4 \font
    }
  } {
    \tl_put_right:Nn \l_fontspec_postadjust_tl {
      \fontdimen 2 \font = #1 \fontdimen 2 \font
      \fontdimen 3 \font = #2 \fontdimen 3 \font
      \fontdimen 4 \font = #3 \fontdimen 4 \font
    }
  }
}

Punctuation space Scaling factor for the nominal \fontdimen7.
\keys_define:nn {fontspec} { PunctuationSpace .code:n = {
  \fontspec_update_fontid:n {+punctspace:#1}
  \tl_put_right:Nx \l_fontspec_postadjust_tl { \fontdimen 7 \font = #1 \fontdimen 7 \font }
}

Secret hook into the font-adjustment code
\keys_define:nn {fontspec} { FontAdjustment .code:n = {
  \fontspec_update_fontid:n {+fontadjust:\detokenize{#1}}
  \tl_put_right:Nx \l_fontspec_postadjust_tl {#1}
}

Letterspacing
\keys_define:nn {fontspec} { LetterSpace .code:n = {
  \fontspec_update_fontid:n {+tracking:#1}
  \fontspec_update_featstr:n{letterspace=#1}
}

Hyphenation character This feature takes one of three arguments: 'None', ⟨glyph⟩, or ⟨slot⟩. If the input isn’t the first, and it’s one character, then it’s the second; otherwise, it’s the third.
Color Hooks into pkgxcolor, which names its colours \color@<name>.

\keys_define:nn {fontspec} { Color .code:n = {
    \fontspec_update_fontid:n {+col:#1}
    \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
        { \convertcolorspec{named}{#1}{HTML}\l_fontspec_hexcol_tl }
        { \int_compare:nTF { \tl_length:n {#1} == 6 } 
            { \tl_set:Nn \l_fontspec_hexcol_tl {#1} } 
            { \int_compare:nTF { \tl_length:n {#1} == 8 } 
                { \fontspec_parse_colour:viii #1 } 
                { \fontspec_warning:nx {bad-colour} {#1} } 
            } 
        } 
    } 
}

\keys_define:nn {fontspec} { HyphenChar .code:n = {
    \fontspec_update_fontid:n {+hyphenchar:#1}
    \str_if_eq:nnTF {#1} {None} 
        { \tl_put_right:Nn \l_fontspec_postadjust_tl 
            { \hyphenchar \font = \c_minus_one } 
        } 
        { \tl_if_single:nTF {#1} 
            { \tl_set:Nn \l_fontspec_hyphenchar_tl {'#1} } 
            { \tl_set:Nn \l_fontspec_hyphenchar_tl { #1} } 
            \font_glyph_if_exist:NnTF \l_fontspec_font {\l_fontspec_hyphenchar_tl} 
                { \tl_put_right:Nn \l_fontspec_postadjust_tl 
                    { ⟨∗xetex⟩
                        { \hyphenchar \font = \l_fontspec_hyphenchar_tl \scan_stop: } 
                    ⟨/xetex⟩ 
                    ⟨∗luatex⟩ 
                        { \hyphenchar \font = \c_zero 
                            \luatexprehyphenchar = \l_fontspec_hyphenchar_tl \scan_stop: 
                        } 
                    ⟨/luatex⟩ 
                } 
                { \fontspec_error:nx {no-glyph} (#1) } 
        } 
    } 
}

93
\cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8 {
  \tl_set:Nn \l_fontspec_hexcol_tl {#1#2#3#4#5#6}
  \tl_if_eq:NNF \l_fontspec_opacity_tl \g_fontspec_opacity_tl {
    \bool_if:NF \l_fontspec_firsttime_bool {
      \fontspec_warning:nx {opa-twice-col} {#7#8}
    }
  }
  \tl_set:Nn \l_fontspec_opacity_tl {#7#8}
}
\aliasfontfeature{Color}{Colour}
\int_new:N \l_fontspec_tmp_int
\keys_define:nn {fontspec} { Opacity .code:n = {
  \fontspec_update_fontid:n {+opac:#1}
  \int_set:Nn \l_fontspec_tmp_int {255}
  \_int_mult_truncate:Nn \l_fontspec_tmp_int { #1 }
  \tl_if_eq:NNF \l_fontspec_opacity_tl \g_fontspec_opacity_tl {
    \bool_if:NF \l_fontspec_firsttime_bool {
      \fontspec_warning:nx {opa-twice} {#1}
    }
  }
  \tl_set:Nx \l_fontspec_opacity_tl {
    \int_compare:nT { \l_fontspec_tmp_int <= "F } {/zero.noslash} % zero pad
    \int_to_hexadecimal:n { \l_fontspec_tmp_int }
  }
}}
\keys_define:nn {fontspec} { Mapping .code:n = {
  \fontspec_update_fontid:n {+map:#1}
  \fontspec_update_featstr:n{mapping=#1}
}}
\Mapping
\keys_define:nn {fontspec} { \mapping \xetexx }
\Mapping
\keys_define:nn {fontspec} { \mapping \luatex }
\Mapping
\keys_define:nn {fontspec} { \mapping \tex-text }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
\keys_define:nn {fontspec} { \mapping \ligtex }
\Mapping
FeatureFile

\keys_define:nn {fontspec} { FeatureFile .code:n = {
  \fontspec_update_fontid:n {+fea:#1}
  \fontspec_update_featstr:n{featurefile=#1}
}

\keys_define:nn {fontspec} { Weight .code:n = {
  \fontspec_update_fontid:n {+weight:#1}
  \fontspec_update_featstr:n{weight=#1}
}

\keys_define:nn {fontspec} { Width .code:n = {
  \fontspec_update_fontid:n {+width:#1}
  \fontspec_update_featstr:n{width=#1}
}

\keys_define:nn {fontspec} { OpticalSize .code:n =
  ⟨∗xetexx⟩{
    \begin_if:NF \l_fontspec_icu_bool {
      \tl_set:Nn \l_fontspec_optical_size_tl {/ S = #1}
      \fontspec_update_fontid:n {+size:#1}
    }{
      \begin_if:N \l_fontspec_mm_bool {
        \fontspec_update_fontid:n {+size:#1}
        \fontspec_update_featstr:n{optical size=#1}
      }{
        \fontspec_warning:n {no-opticals}
      }
    }
  ⟨∗luatex⟩{
    \tl_set:Nn \l_fontspec_optical_size_tl {/ S = #1}
    \fontspec_update_fontid:n {+size:#1}
  }⟨∗luatex⟩

25.6.5 Continuous font axes

\keys_define:nn {fontspec} { Weight .code:n = {
  \fontspec_update_fontid:n {+weight:#1}
  \fontspec_update_featstr:n{weight=#1}
}

\keys_define:nn {fontspec} { Width .code:n = {
  \fontspec_update_fontid:n {+width:#1}
  \fontspec_update_featstr:n{width=#1}
}

\keys_define:nn {fontspec} { OpticalSize .code:n =
  ⟨∗xetexx⟩{
    \begin_if:NF \l_fontspec_icu_bool {
      \tl_set:Nn \l_fontspec_optical_size_tl {/ S = #1}
      \fontspec_update_fontid:n {+size:#1}
    }{
      \begin_if:N \l_fontspec_mm_bool {
        \fontspec_update_fontid:n {+size:#1}
        \fontspec_update_featstr:n{optical size=#1}
      }{
        \fontspec_warning:n {no-opticals}
      }
    }
  ⟨∗luatex⟩{
    \tl_set:Nn \l_fontspec_optical_size_tl {/ S = #1}
    \fontspec_update_fontid:n {+size:#1}
  }⟨∗luatex⟩

25.6.6 Font transformations

These are to be specified to apply directly to a font shape:
These are to be given to a shape that has no real bold/italic to signal that \fontspec should automatically create ‘fake’ shapes.

The behaviour is currently that only if both \AutoFakeSlant and \AutoFakeBold are specified, the bold italic is also faked.

These features presently \emph{override} real shapes found in the font; in the future I’d like these features to be ignored in this case, instead. (This is just a bit harder to program in the current design of \fontspec.)
1726 }\}
1727 },
1728 AutoFakeSlant .default:n = {0.2}
1729 }

Same but reversed:
1730 \keys_define:nn {fontspec} { AutoFakeBold .code:n = {
1731 \bool_if:NT \l_fontspec_firsttime_bool {
1732 \tl_set:Nn \l_fontspec_fake_embolden_tl (#1)
1733 \clist_put_right:Nn \l_fontspec_fontfeat_bf_clist {FakeBold=#1}
1734 \tl_set_eq:NN \l_fontspec_fontname_bf_tl \l_fontspec_fontname_tl
1735 \bool_set_false:N \l_fontspec_nobf_bool
1736 \fontspec_update_fontid:n {fakebf:#1}
1737 \fontspec_update_fontid:n {fakebf:#1}
1738 \fontspec_update_fontid:n {fakebf:#1}
1739 \fontspec_update_fontid:n {fakebf:#1}
1740 \tl_if_empty:NT \l_fontspec_fake_slant_tl {
1741 \clist_put_right:Nx \l_fontspec_fontfeat_bfit_clist {FakeSlant=#1}
1742 \clist_put_right:Nx \l_fontspec_fontfeat_bfit_clist {FakeBold=#1}
1743 \tl_set_eq:NN \l_fontspec_fontname_bfit_tl \l_fontspec_fontname_tl
1744 \fontspec_update_fontid:n {fakebf:#1}
1745 }
1746 }
1747 },
1748 AutoFakeBold .default:n = {1.5}
1749 }

25.6.7 Ligatures

The call to the nested keyval family must be wrapped in braces to hide the parent
list (this later requires the use of global definitions \xdef in [...]). Both AAT and
OpenType names are offered to chose Rare/Discretionary ligatures.
1750 \fontspec_define_font_feature:n(Ligatures)
1751 \fontspec_define_feature_option:nnnnn(Ligatures)(Required) ()(1)(+rlig)
1752 \fontspec_define_feature_option:nnnnn(Ligatures)(NoRequired) ()(1)(+rlig)
1753 \fontspec_define_feature_option:nnnnn(Ligatures)(Common) ()(2)(+liga)
1754 \fontspec_define_feature_option:nnnnn(Ligatures)(NoCommon) ()(3)(-liga)
1755 \fontspec_define_feature_option:nnnnn(Ligatures)(Rare) ()(4)(+dlig)
1756 \fontspec_define_feature_option:nnnnn(Ligatures)(NoRare) ()(5)(-dlig)
1757 \fontspec_define_feature_option:nnnnn(Ligatures)(Discretionary) ()(4)(+dlig)
1758 \fontspec_define_feature_option:nnnnn(Ligatures)(NoDiscretionary)() (5)(-dlig)
1759 \fontspec_define_feature_option:nnnnn(Ligatures)(Contextual) ()(1)(+clig)
1760 \fontspec_define_feature_option:nnnnn(Ligatures)(NoContextual) ()(1)(-clig)
1761 \fontspec_define_feature_option:nnnnn(Ligatures)(Historic) ()(1)(+hlig)
1762 \fontspec_define_feature_option:nnnnn(Ligatures)(NoHistoric) ()(1)(-hlig)
1763 \fontspec_define_feature_option:nnnnn(Ligatures)(Logos) ()(6)( )
1764 \fontspec_define_feature_option:nnnnn(Ligatures)(NoLogos) ()(7)( )
1765 \fontspec_define_feature_option:nnnnn(Ligatures)(Rebus) ()(8)( )
1766 \fontspec_define_feature_option:nnnnn(Ligatures)(NoRebus) ()(9)( )
1767 \fontspec_define_feature_option:nnnnn(Ligatures)(Diphthong) ()(10)( )
1768 \fontspec_define_feature_option:nnnnn(Ligatures)(NoDiphthong) ()(11)( )
1769 \fontspec_define_feature_option:nnnnn(Ligatures)(Squared) ()(12)( )

97
Emulate CM extra ligatures.

\keys_define:nn {fontspec}
{Ligatures / TeX .code:n = {
\fontspec_update_fontid:n {+map:tex-text}
\fontspec_update_featstr:n{mapping=tex-text}
\fontspec_update_fontid:n {+tlig+trep}
\fontspec_update_featstr:n{+tlig;+trep}
}}

25.6.8 Letters

\fontspec_define_font_feature:n{Letters}
\fontspec_define_feature_option:nnnnn{Letters}{Normal} {3}{/zero.noslash}{}
\fontspec_define_feature_option:nnnnn{Letters}{Uppercase} {3}{1}{+case}
\fontspec_define_feature_option:nnnnn{Letters}{Lowercase} {3}{2}{}
\fontspec_define_feature_option:nnnnn{Letters}{SmallCaps} {3}{3}{+smcp}
\fontspec_define_feature_option:nnnnn{Letters}{PetiteCaps} {} {} {+pcap}
\fontspec_define_feature_option:nnnnn{Letters}{UppercaseSmallCaps} {} {} {+c2sc}
\fontspec_define_feature_option:nnnnn{Letters}{UppercasePetiteCaps} {} {} {+c2pc}
\fontspec_define_feature_option:nnnnn{Letters}{InitialCaps} {3}{4}{}
\fontspec_define_feature_option:nnnnn{Letters}{Unicase} {} {} {+unic}
\fontspec_define_feature_option:nnnnn{Letters}{Random} {} {} {+rand}

25.6.9 Numbers

These were originally separated into NumberCase and NumberSpacing following AAT, but it makes more sense to combine them.

Both naming conventions are offered to select the number case.

\fontspec_define_font_feature:n{Numbers}
\fontspec_define_feature_option:nnnnn{Numbers}{Monospaced} {6} {/zero.noslash}{+tnum}
\fontspec_define_feature_option:nnnnn{Numbers}{Proportional} {6} {1}{+pnum}
\fontspec_define_feature_option:nnnnn{Numbers}{OldStyle} {21}{0}{+omonum}
\fontspec_define_feature_option:nnnnn{Numbers}{Lining} {21}{1}{+lnum}
\fontspec_define_feature_option:nnnnn{Numbers}{SlashedZero} {14}{5}{+zero}
\fontspec_define_feature_option:nnnnn{Numbers}{NoSlashedZero}{14}{4}{-zero}

luaotload provides a custom anum feature for replacing Latin (AKA Arabic) numbers with Arabic (AKA Indic-Arabic). The same feature maps to Farsi (Persian) numbers if font language is Farsi.

\luatex_if_engine:T { 
\fontspec_define_feature_option:nnnnn{Numbers}{Arabic}{}{}{+anum}
}
25.6.10  Contextuals

\fontspec_define_font_feature:n {Contextuals}
\fontspec_define_feature_option:nnnnn{Contextuals}{Swash} {} {} {+cswh}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoSwash} {} {} {-cswh}
\fontspec_define_feature_option:nnnnn{Contextuals}{Alternate} {} {} {+calt}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoAlternate} {} {} {-calt}
\fontspec_define_feature_option:nnnnn{Contextuals}{WordInitial} {8}{/zero.noslash}{+init}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoWordInitial}{8}{1}{-init}
\fontspec_define_feature_option:nnnnn{Contextuals}{WordFinal} {8}{2}{+fina}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoWordFinal} {8}{3}{-fina}
\fontspec_define_feature_option:nnnnn{Contextuals}{LineInitial} {8}{4}{+falt}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoLineInitial}{8}{5}{-falt}
\fontspec_define_feature_option:nnnnn{Contextuals}{LineFinal} {8}{6}{+medi}
\fontspec_define_feature_option:nnnnn{Contextuals}{NoLineFinal} {8}{7}{-medi}

25.6.11  Diacritics

\fontspec_define_font_feature:n{Diacritics}
\fontspec_define_feature_option:nnnnn{Diacritics}{Show} {9}{/zero.noslash}{}
\fontspec_define_feature_option:nnnnn{Diacritics}{Hide} {9}{1}{}
\fontspec_define_feature_option:nnnnn{Diacritics}{Decompose} {9}{2}{}
\fontspec_define_feature_option:nnnnn{Diacritics}{MarkToBase} {9}{3}{+mark}
\fontspec_define_feature_option:nnnnn{Diacritics}{NoMarkToBase} {9}{4}{-mark}
\fontspec_define_feature_option:nnnnn{Diacritics}{MarkToMark} {9}{5}{+mkmk}
\fontspec_define_feature_option:nnnnn{Diacritics}{NoMarkToMark} {9}{6}{-mkmk}
\fontspec_define_feature_option:nnnnn{Diacritics}{AboveBase} {9}{7}{+abvm}
\fontspec_define_feature_option:nnnnn{Diacritics}{NoAboveBase} {9}{8}{-abvm}
\fontspec_define_feature_option:nnnnn{Diacritics}{BelowBase} {9}{9}{+blwm}
\fontspec_define_feature_option:nnnnn{Diacritics}{NoBelowBase} {9}{10}{-blwm}

25.6.12  Kerning

\fontspec_define_font_feature:n{Kerning}
\fontspec_define_feature_option:nnnnn{Kerning}{Uppercase} {10}{/zero.noslash}{+cpsp}
\fontspec_define_feature_option:nnnnn{Kerning}{On} {10}{1}{+kern}
\fontspec_define_feature_option:nnnnn{Kerning}{Off} {10}{2}{-kern}
\fontspec_define_feature_option:nnnnn{Kerning}{Vertical} {10}{3}{+vkrn}
\fontspec_define_feature_option:nnnnn{Kerning}{VerticalAlternateProportional} {10}{4}{+vpal}
\fontspec_define_feature_option:nnnnn{Kerning}{VerticalAlternateHalfWidth} {10}{5}{+vhal}

25.6.13  Vertical position

\fontspec_define_font_feature:n{VerticalPosition}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Normal} {10}{0}{}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Superior} {10}{1}{+sups}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Inferior} {10}{2}{+subs}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Ordinal} {10}{3}{+ordn}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Numerator} {10}{4}{+numr}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{Denominator} {10}{5}{+dnom}
\fontspec_define_feature_option:nnnnn{VerticalPosition}{ScientificInferior} {10}{6}{+sinf}
25.6.14 Fractions

\fontspec_define_font_feature:n{Fractions}
\fontspec_define_feature_option:nnnnn{Fractions}{On} {11}{1}{+frac}
\fontspec_define_feature_option:nnnnn{Fractions}{Off} {11}{0}{-frac}
\fontspec_define_feature_option:nnnnn{Fractions}{Diagonal} {11}{2}{}
\fontspec_define_feature_option:nnnnn{Fractions}{Alternate}{} {} {+afrc}

25.6.15 Alternates and variants

Selected numerically because they don’t have standard names. Very easy to process, very annoying for the user!

\fontspec_define_font_feature:n { Alternate }
\keys_define:nn {fontspec}{
    Alternate .default:n = {/zero.noslash} ,
    Alternate / unknown .code:n =
    {
        \clist_map_inline:nn {#1}
        { \fontspec_make_feature:nnx {17}{##1} { \fontspec_salt:n {##1} } }
    }
}
\cs_set:Nn \fontspec_salt:n
\clist_map_inline:nn {#1}{ +salt = #1 }
\clist_map_inline:nn {#1}{ +salt = \int_eval:n {#1+1} }
\fontspec_define_font_feature:n { Variant }
\keys_define:nn {fontspec}{
    Variant .default:n = {/zero.noslash} ,
    Variant / unknown .code:n =
    {
        \clist_map_inline:nn {##1}
        { \fontspec_make_feature:nnx {18}{##1} { +ss \two@digits {##1} } }
    }
}
\aliasfontfeature{Variant}{StylisticSet}
\fontspec_define_font_feature:n { CharacterVariant }
\use:x
\cs_new:Npn \exp_not:N \fontspec_parse_cv:w
    ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
    { \fontspec_make_numbered_feature:xn
        +cv \exp_not:N \two@digits {##1} }{##2}
\keys_define:nn {fontspec}{
    CharacterVariant / unknown .code:n =
    {
        \clist_map_inline:nn {##1}{#1}
    }
}

Possibilities: a:₀:_initial or a:₁:_initial.

### 25.6.16 OpenType maths font features

Deprecated August 2011; delete at some stage in the future.

```latex
\keys_define:nn {fontspec}
  {ScriptStyle .code:n = {
    ⟨xetex⟩ \fontspec_update_fontid:n {+ssty=0}
    ⟨luatex⟩ \fontspec_update_fontid:n (+ssty=1) \fontspec_update_featstr:n{+sstyle}}
  },
  ScriptScriptStyle .code:n = {
    ⟨xetex⟩ \fontspec_update_fontid:n {+ssty=1}
    ⟨luatex⟩ \fontspec_update_fontid:n {+ssty=2} \fontspec_update_featstr:n{+ssstyle}
  }
}
```

### 25.6.17 Style

```latex
\fontspec_define_font_feature:n{Style}
\fontspec_define_feature_option:nnnnn{Style}{Alternate} {} {} {+salt}
\fontspec_define_feature_option:nnnnn{Style}{Italic} {32}{2}{+ital}
\fontspec_define_feature_option:nnnnn{Style}{Ruby} {28}{2}{+ruby}
\fontspec_define_feature_option:nnnnn{Style}{Swash} {} {} {+swsh}
\fontspec_define_feature_option:nnnnn{Style}{Historic} {} {} {+hist}
\fontspec_define_feature_option:nnnnn{Style}{Display} {19}{1}{}
\fontspec_define_feature_option:nnnnn{Style}{Engraved} {19}{2}{}
\fontspec_define_feature_option:nnnnn{Style}{TitlingCaps} {19}{4}{} {+titl}
\fontspec_define_feature_option:nnnnn{Style}{TallCaps} {19}{5}{}
\fontspec_define_feature_option:nnnnn{Style}{HorizontalKana} {} {} {+hkna}
\fontspec_define_feature_option:nnnnn{Style}{VerticalKana} {} {} {+vkna}
\fontspec_define_numbered_feat:nnnn {Style} {MathScript} {MathScriptScript} {+ssty} {0}
```

### 25.6.18 CJK shape

```latex
\fontspec_define_font_feature:n{CJKShape}
\fontspec_define_feature_option:nnnnn{CJKShape}{Traditional}{}{}{+trad}
\fontspec_define_feature_option:nnnnn{CJKShape}{Simplified}{}{}{+smpl}
\fontspec_define_feature_option:nnnnn{CJKShape}{JIS1978}{}{}{+jp78}
\fontspec_define_feature_option:nnnnn{CJKShape}{JIS1983}{}{}{+jp83}
\fontspec_define_feature_option:nnnnn{CJKShape}{JIS1990}{}{}{+jp90}
\fontspec_define_feature_option:nnnnn{CJKShape}{Expert}{}{}{+expt}
\fontspec_define_feature_option:nnnnn{CJKShape}{NLC}{}{}{+nlck}
```
25.6.19 Character width
\fontspec_define_font_feature:n{CharacterWidth}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Proportional}{22}{0}{+pwid}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Full}{22}{1}{+fvid}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Half}{22}{2}{+hvid}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Third}{22}{3}{+twid}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Quarter}{22}{4}{+qwid}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{AlternateProportional}{22}{5}{+palt}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{AlternateHalf}{22}{6}{+halt}
\fontspec_define_feature_option:nnnnn{CharacterWidth}{Default}{22}{7}{

25.6.20 Annotation
\fontspec_define_feature_option:nnnnn{Annotation}{Off}{24}{0}{}
\fontspec_define_feature_option:nnnnn{Annotation}{Box}{24}{1}{}
\fontspec_define_feature_option:nnnnn{Annotation}{RoundedBox}{24}{2}{}
\fontspec_define_feature_option:nnnnn{Annotation}{Circle}{24}{3}{}
\fontspec_define_feature_option:nnnnn{Annotation}{BlackCircle}{24}{4}{}
\fontspec_define_feature_option:nnnnn{Annotation}{Parenthesis}{24}{5}{}
\fontspec_define_feature_option:nnnnn{Annotation}{Period}{24}{6}{}
\fontspec_define_feature_option:nnnnn{Annotation}{RomanNumerals}{24}{7}{}
\fontspec_define_feature_option:nnnnn{Annotation}{Diamond}{24}{8}{}
\fontspec_define_feature_option:nnnnn{Annotation}{BlackSquare}{24}{9}{}
\fontspec_define_feature_option:nnnnn{Annotation}{BlackRoundSquare}{24}{10}{}
\fontspec_define_feature_option:nnnnn{Annotation}{DoubleCircle}{24}{11}{}
\keys_define:nn {fontspec}
{ Annotation .default:n = {/zero.noslash} ,
Annotation / unknown .code:n =
{\fontspec_make_feature:nnx {}{}}
{\xetexx \int_eval:n {#1+1} }
{\luatex \int_eval:n (#1+1) }
}
25.6.21 Vertical
\keys_define:nn {fontspec}
{ Vertical .choice: ,
Vertical / RotatedGlyphs .code:n =
{\bool_if:NTF \l_fontspec_icu_bool {
\fontspec_make_feature:nnn{}{}{+vrt2}
\fontspec_update_fontid:n (+vert)
\fontspec_update_featstr:n(vertical)
}
{\fontspec_update_fontid:n (+vert)
\fontspec_update_featstr:n(vertical)
}}
\newfontscript{Arabic}{arab} \newfontscript{Armenian}{armn}
\newfontscript{Balinese}{bali} \newfontscript{Bengali}{beng}
\newfontscript{Bopomofo}{bopo} \newfontscript{Braille}{brai}
\newfontscript{Buginese}{bugi} \newfontscript{Buhid}{buhd}
\newfontscript{Byzantine\textsuperscript{Music}}{byzm}
\newfontscript{Canadian\textsuperscript{Syllabics}}{cans}
\newfontscript{Cherokee}{cher}
\newfontscript{CJK\textsuperscript{Ideographic}}{hani} \newfontscript{Coptic}{copt}
\newfontscript{Cypr	extsuperscript{i}otic\textsuperscript{Syllabary}}{cprt} \newfontscript{Cyrillic}{cyr1}
\newfontscript{Default}{DFLT} \newfontscript{Deseret}{dsrt}
\newfontscript{Devanagari}{deva} \newfontscript{Ethiopic}{ethi}
\newfontscript{Georgian}{geor} \newfontscript{Glagolitic}{glag}
\newfontscript{Gothic}{goth} \newfontscript{Greek}{grek}
\newfontscript{Gujarati}{gujr} \newfontscript{Gurmukhi}{guru}
\newfontscript{Hangul\textsuperscript{Jamo}}{jamo} \newfontscript{Hangul}{hang}
\newfontscript{Hanunoo}{hano} \newfontscript{Hebrew}{hebr}
\newfontscript{Hiragana\textsuperscript{and}Katakana}{kana}
\newfontscript{Javanese}{java} \newfontscript{Kannada}{knda}
\newfontscript{Kharosthi}{khar} \newfontscript{Khmer}{khmr}
\newfontscript{Lao}{lao} \newfontscript{Latin}{latn}
\newfontscript{Limbu}{limb} \newfontscript{Linear\textsuperscript{B}}{linb}
\newfontscript{Malayalam}{mlym} \newfontscript{Math}{math}
\newfontscript{Mongolian}{mong} \newfontscript{Myanmar}{mymr}
\newfontscript{N\textsuperscript{\text{\textbackslash{}ko}}}{nko} \newfontscript{Ogham}{ogam}
\newfontscript{Old\textsuperscript{\text{\textbackslash{}Italic}}}{ital}
\newfontscript{Old\textsuperscript{\text{\textbackslash{}Persian}}\textsuperscript{\textsuperscript{\textbackslash{}Cuneiform}}}{xpeo}
\newfontscript{Oriya}{orya} \newfontscript{Osmanya}{osma}
\newfontscript{Phags-pa}{phag} \newfontscript{Phoenician}{phnx}
\newfontscript{Runic}{runr} \newfontscript{Shavian}{shaw}
\newfontscript{Sinhala}{sinh} \newfontscript{Sumerian\textsuperscript{\text{\textbackslash{}Akadian}}\textsuperscript{\textsuperscript{\textbackslash{}Cuneiform}}}{xsux}
\newfontscript{Syriac}{syrc}
\newfontscript{Tagalog}{tlgl}
\newfontscript{Tai\textsuperscript{\text{\textbackslash{}Le}}}{tale}
\newfontscript{Tamil}{taml}
\newfontscript{Telugu}{telu}
\newfontscript{Thai}{thai}
\newfontscript{Tibetan}{tibt}
\newfontscript{Tifinagh}{tfng}
\newfontscript{Ugaritic\textsuperscript{\text{\textbackslash{}Cuneiform}}}{ugar}

\newfontlanguage{Abaza}{ABA}\newfontlanguage{Abkhazian}{ABK}
\newfontlanguage{Adyghe}{ADY}\newfontlanguage{Afrikaans}{AFK}
\newfontlanguage{Afar}{AFR}\newfontlanguage{Agaw}{AGW}

For convenience:
\newfontscript{Kana}{kana} \newfontscript{Maths}{math}
\newfontscript{CJK}{hani}

\textbf{25.6.23 Language}
<p>| 2027 | newfontlanguage{Altai}{ALT} | newfontlanguage{Amharic}{AMH} |
| 2028 | newfontlanguage{Arabic}{ARA} | newfontlanguage{Aari}{ARI} |
| 2029 | newfontlanguage{Arakanese}{ARK} | newfontlanguage{Assamese}{ASM} |
| 2030 | newfontlanguage{Athapaskan}{ATH} | newfontlanguage{Avar}{AVR} |
| 2031 | newfontlanguage{Awadhi}{AWA} | newfontlanguage{Aymara}{AYM} |
| 2032 | newfontlanguage{Azeri}{AZE} | newfontlanguage{Badaga}{BAD} |
| 2033 | newfontlanguage{Baghelkhandi}{BAG} | newfontlanguage{Balkar}{BAL} |
| 2034 | newfontlanguage{Baule}{BAU} | newfontlanguage{Berber}{BBR} |
| 2035 | newfontlanguage{Bench}{BCH} | newfontlanguage{Bible ‘Cree}{BCL} |
| 2036 | newfontlanguage{Belarussian}{BEL} | newfontlanguage{Bemba}{BEM} |
| 2037 | newfontlanguage{Bengali}{BEN} | newfontlanguage{Bulgarian}{BGR} |
| 2038 | newfontlanguage{Bhili}{BHI} | newfontlanguage{Bhojpuri}{BHO} |
| 2039 | newfontlanguage{Bikol}{BIK} | newfontlanguage{Bilin}{BIL} |
| 2040 | newfontlanguage{Blackfoot}{BKF} | newfontlanguage{Balochi}{BLI} |
| 2041 | newfontlanguage{Balante}{BLN} | newfontlanguage{Balte}{BLT} |
| 2042 | newfontlanguage{Bambara}{BMB} | newfontlanguage{Bamileke}{BML} |
| 2043 | newfontlanguage{Breton}{BRE} | newfontlanguage{Brahui}{BRI} |
| 2044 | newfontlanguage{Braj ‘Bhasha}{BRI} | newfontlanguage{Burmese}{BRM} |
| 2045 | newfontlanguage{Bashkir}{BSH} | newfontlanguage{Beti}{BTI} |
| 2046 | newfontlanguage{Catalan}{CAT} | newfontlanguage{Cebuano}{CEB} |
| 2047 | newfontlanguage{Chechen}{CHE} | newfontlanguage{Chaha ‘Gurage}{CHG} |
| 2048 | newfontlanguage{Chattisgarhi}{CHH} | newfontlanguage{Chichewa}{CHI} |
| 2049 | newfontlanguage{Chukchi}{CHK} | newfontlanguage{Chipewyan}{CHP} |
| 2050 | newfontlanguage{Cherokee}{CHR} | newfontlanguage{Chuvash}{CHV} |
| 2051 | newfontlanguage{Comorian}{CMR} | newfontlanguage{Coptic}{COP} |
| 2052 | newfontlanguage{Cree}{CRE} | newfontlanguage{Carrier}{CRR} |
| 2053 | newfontlanguage{Crimean Tatar}{CRT} | newfontlanguage{Church ‘Slavonic}{CSL} |
| 2054 | newfontlanguage{Czech}{CZ} | newfontlanguage{Danish}{DAN} |
| 2055 | newfontlanguage{Dargwa}{DAR} | newfontlanguage{Woods ‘Cree}{DCR} |
| 2056 | newfontlanguage{German}{DEU} | |
| 2057 | newfontlanguage{Djargi}{DJR} | newfontlanguage{Divish}{DIV} |
| 2058 | newfontlanguage{Djerma}{DJR} | newfontlanguage{Dangme}{DNG} |
| 2059 | newfontlanguage{Dinka}{DK} | newfontlanguage{Dungan}{DUN} |
| 2060 | newfontlanguage{Dzongkha}{DZM} | newfontlanguage{Ebira}{EBI} |
| 2061 | newfontlanguage{Eastern ‘Cree}{ECR} | newfontlanguage{Edo}{EDO} |
| 2062 | newfontlanguage{Efik}{EF} | newfontlanguage{Greek}{ELL} |
| 2063 | newfontlanguage{English}{ENG} | newfontlanguage{Erzya}{ERZ} |
| 2064 | newfontlanguage{Spanish}{ESP} | newfontlanguage{Esperanto}{ET} |
| 2065 | newfontlanguage{Basque}{EUQ} | newfontlanguage{Evenki}{EVK} |
| 2066 | newfontlanguage{Even}{EVN} | newfontlanguage{Ewe}{EWE} |
| 2067 | newfontlanguage{French}{FRA} | newfontlanguage{Französisch}{FRA} |
| 2068 | newfontlanguage{Farsi}{FAR} | newfontlanguage{Parsi}{FA} |
| 2069 | newfontlanguage{Persian}{FA} | |
| 2070 | newfontlanguage{Finnish}{FIN} | newfontlanguage{Fijian}{FJI} |
| 2071 | newfontlanguage{Flemish}{FLE} | newfontlanguage{Forest ‘Nenets}{FNE} |
| 2072 | newfontlanguage{Fon}{FON} | newfontlanguage{Faroese}{FOS} |
| 2073 | newfontlanguage{French}{FRA} | newfontlanguage{Friulian}{FRI} |
| 2074 | newfontlanguage{Friulian}{FRI} | newfontlanguage{Futa}{FTA} |
| 2075 | newfontlanguage{Fulani}{FUL} | newfontlanguage{Ga}{GAD} |
| 2076 | newfontlanguage{Gaelic}{GAE} | newfontlanguage{Gagauz}{GAG} |</p>
<table>
<thead>
<tr>
<th>Font Language</th>
<th>Font Language</th>
<th>Font Language</th>
<th>Font Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galician</td>
<td>Garshuni</td>
<td>Ge'ez</td>
<td>Greenlandic</td>
</tr>
<tr>
<td>Garhwali</td>
<td>Gumuz</td>
<td>Guma</td>
<td>Garhwali</td>
</tr>
<tr>
<td>Gondi</td>
<td>Greenlanderic</td>
<td>Guaranji</td>
<td>Guaranji</td>
</tr>
<tr>
<td>Gro</td>
<td>Guam</td>
<td>Harari</td>
<td>Hausa</td>
</tr>
<tr>
<td>Hammer-Banna</td>
<td>Hiligaynon</td>
<td>Hilo</td>
<td>Hilo</td>
</tr>
<tr>
<td>Hindi</td>
<td>High Mari</td>
<td>Ho</td>
<td>Hmong</td>
</tr>
<tr>
<td>Hindi</td>
<td>High Mari</td>
<td>Ho</td>
<td>Hmong</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Croatian</td>
<td>Hrv</td>
<td>Hrv</td>
</tr>
<tr>
<td>Igbo</td>
<td>Ijo</td>
<td>Ilo</td>
<td>Ind</td>
</tr>
<tr>
<td>Ilokano</td>
<td>Indonesian</td>
<td>Inari Sami</td>
<td>Inari Sami</td>
</tr>
<tr>
<td>Icelandic</td>
<td>Italian</td>
<td>Hebrew</td>
<td>Hebrew</td>
</tr>
<tr>
<td>Javanese</td>
<td>Yiddish</td>
<td>Jii</td>
<td>Jii</td>
</tr>
<tr>
<td>Japanese</td>
<td>Judezmo</td>
<td>Jud</td>
<td>Jud</td>
</tr>
<tr>
<td>Jula</td>
<td>Kalenjin</td>
<td>Kal</td>
<td>Kal</td>
</tr>
<tr>
<td>Jurchi</td>
<td>Kalenjin</td>
<td>Kal</td>
<td>Kal</td>
</tr>
<tr>
<td>Jurchi</td>
<td>Karachay</td>
<td>Karachay</td>
<td>Karachay</td>
</tr>
<tr>
<td>Karakul</td>
<td>Kazak</td>
<td>KA2</td>
<td>KA2</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Kebena</td>
<td>Kazakh</td>
<td>Kha</td>
<td>Kha</td>
</tr>
<tr>
<td>Font Language</td>
<td>ISO Language Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lingala</td>
<td>LIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-Mari</td>
<td>LMA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limbu</td>
<td>LMB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lomwe</td>
<td>LMW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower-Sorbian</td>
<td>LSB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lule-Sami</td>
<td>LSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lithuanian</td>
<td>LTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luba</td>
<td>LUB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luganda</td>
<td>LUG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luha</td>
<td>LUK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latvian</td>
<td>LVI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Majang</td>
<td>MAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malayalam-Traditional</td>
<td>MAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marathi</td>
<td>MAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marwari</td>
<td>MAW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mbundu</td>
<td>MBN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manchu</td>
<td>MCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moose-Cree</td>
<td>MCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mende</td>
<td>MDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Me'en</td>
<td>MEN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mizo</td>
<td>MIZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macedonian</td>
<td>MKD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>MLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malagasy</td>
<td>MLG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malinke</td>
<td>MLN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malayalam-Reformed</td>
<td>MLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malay</td>
<td>MLY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mandarin</td>
<td>MND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mongolian</td>
<td>MNG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manipuri</td>
<td>MNI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maninka</td>
<td>MNK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manx-Gaelic</td>
<td>MNX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moksha</td>
<td>MOK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moldavian</td>
<td>MOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon</td>
<td>MON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morocan</td>
<td>MOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maori</td>
<td>MRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mountini</td>
<td>MTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malay</td>
<td>MLT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naga-Asamese</td>
<td>NAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nana</td>
<td>NAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naskapi</td>
<td>NAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ndebele</td>
<td>NDB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ndonga</td>
<td>NDG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nepal</td>
<td>NEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Niau</td>
<td>NIA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nis</td>
<td>NIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nkole</td>
<td>NKL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dutch</td>
<td>NLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noraj</td>
<td>NOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northern-Sami</td>
<td>NSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nynorsk</td>
<td>NYN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ojibway</td>
<td>OJB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oromoo</td>
<td>ORO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ossetian</td>
<td>OSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Palestinian-Aramaic</td>
<td>PAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panjabi</td>
<td>PAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pashto</td>
<td>PAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polytonic-Greek</td>
<td>PGL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polish</td>
<td>PLK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provencal</td>
<td>PRO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portuguese</td>
<td>PTG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rajasthani</td>
<td>RAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Russian-Buriat</td>
<td>RBU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rheto-Romanic</td>
<td>RMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romanian</td>
<td>ROM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romany</td>
<td>ROY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Russian</td>
<td>RUS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sadri</td>
<td>SAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sanskrit</td>
<td>SAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Santali</td>
<td>SAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sekota</td>
<td>SEK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selkup</td>
<td>SEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sango</td>
<td>SGO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shan</td>
<td>SHN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sibe</td>
<td>SIB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sidamo</td>
<td>SID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Silte-Gurage</td>
<td>SIG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
 Turkish Turns out that many fonts use ‘TUR’ as their Turkish language tag rather than the specified ‘TRK’. So we check for both:
\*{\setint{\fontspec_language_int}{\fontspec_strnum_int}}
\*{\fontspec_update_fontid:n \{+lang=Turkish\}}
\*{\setint{\fontspec_language_int}{\fontspec_lang_tl}{TUR}}
\*{\fontspec_warning:nx \{language-not-exist\} \{Turkish\}}
\*{\keys_set:nn \{fontspec\} \{Language=Default\}}
\*{\keys_set:nn \{fontspec\} \{Language=Default\}}

\textbf{Default}
\keys_define:nn {fontspec}
{RawFeature .code:n =}
{\fontspec_update_fontid:n \{+Raw:#1\}}
{\fontspec_update_featstr:n[#1]}

\textbf{25.6.24 Raw feature string}
This allows savvy \TeX{}-ers to input font features manually if they have already memorised the OpenType abbreviations and don’t mind not having error checking.
\keys_define:nn {fontspec}
{RawFeature .code:n =}
{\fontspec_update_fontid:n \{+Raw:#1\}}
{\fontspec_update_featstr:n[#1]}

\textbf{25.7 Italic small caps}
The following code for utilising italic small caps sensibly is inspired from Philip Lehman’s \textit{The Font Installation Guide}. Note that \texttt{\upshape} needs to be used \textit{twice} to get from italic small caps to regular upright (it always goes to small caps, then regular upright).

First, the commands for actually selecting italic small caps are defined. I use \texttt{si} as the NFSS shape for italic small caps, but I have seen \texttt{itsc} and \texttt{s1sc} also used. \texttt{\sidefault} may be redefined to one of these if required for compatibility.

\providecommand*{\sidefault}{si}
\DeclareRobustCommand{\sishape}{
\not@math@alphabet\sishape\relax
\fontshape\sidefault\selectfont}

\texttt{\sishape}
This is the macro which enables the overload on the \..shape commands. It takes three such arguments. In essence, the macro selects the first argument, unless the second argument is already selected, in which case it selects the third.

\cs_new:Nn \fontspec_blend_shape:nnn { 
\bool_if:nTF 
{ \str_if_eq_p:xx {\f@shape} {#2} && 
\cs_if_exist_p:c {\f@encoding/\f@family/\f@series/#3} } 
{ \fontshape{#3}\selectfont } 
{ \fontshape{#1}\selectfont } 
}

Here the original \..shape commands are redefined to use the merge shape macro.

\DeclareRobustCommand \itshape { 
\not@math@alphabet\itshape\mathit 
\fontspec_blend_shape:nnn\itdefault\scdefault\sidefault 
}

\DeclareRobustCommand \slshape { 
\not@math@alphabet\slshape\relax 
\fontspec_blend_shape:nnn\sldefault\scdefault\sidefault 
}

\DeclareRobustCommand \scshape { 
\not@math@alphabet\scshape\relax 
\fontspec_blend_shape:nnn\scdefault\itdefault\sidefault 
}

\DeclareRobustCommand \upshape { 
\not@math@alphabet\upshape\relax 
\fontspec_blend_shape:nnn\updefault\sidefault\scdefault 
}

25.8 Selecting maths fonts

Here, the fonts used in math mode are redefined to correspond to the default roman, sans serif and typewriter fonts. Unfortunately, you can only define maths fonts in the preamble, otherwise I’d run this code whenever \setmainfont and friends was run.

\fontspec_setup_maths: Everything here is performed \AtBeginDocument in order to overwrite euler’s attempt. This means fontspec must be loaded after euler. We set up a conditional to return an error if this rule is violated.

Since every maths setup is slightly different, we also take different paths for defining various math glyphs depending which maths font package has been loaded.

\ifpackageloaded(euler){ 
\bool_set_true:N \g_fontspec_package_euler_loaded_bool 
}
\bool_set_false:N \g_fontspec_package_euler_loaded_bool
Knuth’s CM fonts fonts are all squashed together, combining letters, accents, text symbols and maths symbols all in the one font, \texttt{cmr}, plus other things in other fonts. Because we are changing the roman font in the document, we need to redefine all of the maths glyphs in \LaTeX’s operators maths font to still go back to the legacy \texttt{cmr} font for all these random glyphs, unless a separate maths font package has been loaded instead.

In every case, the maths accents are always taken from the operators font, which is generally the main text font. (Actually, there is a \texttt{hat} accent in \texttt{EulerFraktur}, but it’s ugly. So I ignore it. Sorry if this causes inconvenience.)

\texttt{\colon: what’s going on?}  Okay, so \texttt{\colon} in maths mode are defined in a few places, so I need to work out what does what. Respectively, we have:

\begin{verbatim}
% fontmath.ltx:
\DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
\DeclareMathSymbol{\mathpunct}{\mathrel}{operators}{"3A}

% amsmath.sty:
\renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}\nonscript
\mkern-\thinmuskip\cdot\mskip6muplus1mu\relax}

% euler.sty:
\DeclareMathSymbol{\mathrel}{\mathalpha}{EulerFraktur}{"3A}
\DeclareMathSymbol{\mathrel}{\mathalpha}{EulerFraktur}{"3A}

% lucbmath.sty:
\end{verbatim}
So I think, based on this summary, that it is fair to tell fontspec to 'replace' the operators font with legacymaths for this symbol, except when amsmath is loaded since we want to keep its definition.

The following symbols are only defined specifically in euler, so skip them if that package is loaded.

And these ones are defined both in euler and lucbmath, so we only need to run this code if no extra maths package has been loaded.
Finally, we change the font definitions for \textsc{mathrm} and so on. These are defined using the \texttt{\_g\_fontspec\_mathrm\_tl} (...) macros, which default to \texttt{\rmdefault} but may be specified with the \texttt{\setmathrm{...}} commands in the preamble.

Since \LaTeX{} only generally defines one level of boldness, we omit \texttt{\textbf{maths}} in the bold maths series. It can be specified as per usual with \texttt{\setbmathrm}, which stores the appropriate family name in \texttt{\_g\_fontspec\_bfmathrm\_tl}.

\begin{verbatim}
\DeclareSymbolFont{operators}{\g_fontspec_encoding_tl} \g_fontspec_mathrm_tl \mddefault \updefault
\SetSymbolFont{operators}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \updefault
\SetMathAlphabet{\mathrm}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \updefault
\SetMathAlphabet{\mathit}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \itdefault
\SetMathAlphabet{\mathbf}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \updefault
\SetMathAlphabet{\mathsf}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathsf_tl} \mddefault \updefault
\SetMathAlphabet{\mathtt}{normal} {\g_fontspec_encoding_tl} {\g_fontspec_mathtt_tl} \mddefault \updefault
\tl_if_empty:NTF \g_fontspec_bfmathrm_tl {
  \SetMathAlphabet{\mathrm}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \updefault
  \SetMathAlphabet{\mathit}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_mathrm_tl} \mddefault \itdefault
}{
  \SetMathAlphabet{\mathrm}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_bfmathrm_tl} \mddefault \updefault
  \SetMathAlphabet{\mathit}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_bfmathrm_tl} \mddefault \itdefault
}
\SetMathAlphabet{\mathsf}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_mathsf_tl} \mddefault \updefault
\SetMathAlphabet{\mathtt}{bold} {\g_fontspec_encoding_tl} {\g_fontspec_mathtt_tl} \mddefault \updefault
\end{verbatim}

We're a little less sophisticated about not executing the maths setup if various other maths font packages are loaded. This list is based on the wonderful \LaTeX{} Font Catalogue: \url{http://www.tug.dk/FontCatalogue/mathfonts.html}. I'm sure there are more I've missed. Do the \TeX{} Gyre fonts have maths support yet?

Untested: would \texttt{\unless\ifnum\Gamma=28672\relax\bool_set_false:N \g_fontspec_math_bool\fi} be a better test? This needs more cooperation with euler and lucida, I think.

\begin{verbatim}
cs_new:Nn \fontspec\_maybe\_setup\_maths: { 
  \ifpackage\loaded{anntor}{
    \def\mathversions{} \bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{arev}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{eulervm}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{mathdesign}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{concmath}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{cmbright}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{mathesf}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{gfsartemisia}{\bool_set_false:N \g_fontspec_math_bool\fi}
  \ifpackage\loaded{gfsneohellenic}{\bool_set_false:N \g_fontspec_math_bool\fi}
}\end{verbatim}
2388 \@ifpackageloaded{iwona}{
2389 \ifx\define@iwona@mathversions a\bool_set_false:N \g_fontspec_math_bool\fi}
2390 \@ifpackageloaded{kpfonts}{\bool_set_false:N \g_fontspec_math_bool}{}
2391 \@ifpackageloaded{kmath}{\bool_set_false:N \g_fontspec_math_bool}{}
2392 \@ifpackageloaded{kurier}{
2393 \ifx\define@kurier@mathversions a\bool_set_false:N \g_fontspec_math_bool\fi}{}
2394 \@ifpackageloaded{fouriernc}{\bool_set_false:N \g_fontspec_math_bool}{}
2395 \@ifpackageloaded{fourier}{\bool_set_false:N \g_fontspec_math_bool}{}
2396 \@ifpackageloaded{lmodern}{\bool_set_false:N \g_fontspec_math_bool}{}
2397 \@ifpackageloaded{mathpazo}{\bool_set_false:N \g_fontspec_math_bool}{}
2398 \@ifpackageloaded{mathptmx}{\bool_set_false:N \g_fontspec_math_bool}{}
2399 \@ifpackageloaded{MinionPro}{\bool_set_false:N \g_fontspec_math_bool}{}
2400 \@ifpackageloaded{unicode-math}{\bool_set_false:N \g_fontspec_math_bool}{}
2401 \@ifpackageloaded{breqn}{\bool_set_false:N \g_fontspec_math_bool}{}
2402 \bool_if:NT \g_fontspec_math_bool {
2403 \fontspec_info:n {setup-math}
2404 \fontspec_setup_maths:
2405 }
2406 }
2407 \AtBeginDocument{$\texttt{fontspec\_maybe\_setup\_maths}$}

25.9 Finishing up

Now we just want to set up loading the .cfg file, if it exists.

\begin{verbatim}
2408 \bool_if:NT \g_fontspec_cfg_bool { \InputIfFileExists{fontspec.cfg} \}
2410 \typeout{No \textasciitilde fontspec.cfg \textasciitilde file\textasciitilde found; \textasciitilde no\textasciitilde configuration\textasciitilde loaded.})
2412 }

25.10 Compatibility

Old interfaces. These are needed by, at least, the mathspec package.

\begin{verbatim}
2413 \zl_set:Nn \zf@enc { \g_fontspec_encoding_tl }
2414 \zl@basefont \cs_set:Npn \zf@fontspec \#1 \#2
2415 \zl@fontspec \zl@fontspec_select:nn \zl@family \zl@fontspec_family_tl
2417 \zl@set:Nn \zf@family \zl@fontspec_family_tl
2418 \zl@set:Nn \zf@basefont \zl@fontspec_font
2419 }

The end! Thanks for coming.
\end{verbatim}

\ExplSyntaxOff

\@fontspec & \{xetex | luatex\}
First we define some metadata.

```lua
fontspec = { }
fontspec.module = {
  name = "fontspec",
  version = 2.0,
  date = "2009/12/04",
  description = "Advanced font selection for LuaLaTeX.",
  author = "Khaled Hosny",
  copyright = "Khaled Hosny",
  license = "LPPL"
}

local err, warn, info, log = luatexbase.provides_module(fontspec.module)
```

Some utility functions

```lua
fontspec.log = log
fontspec.warning = warn
fontspec.error = err
```

The following functions check for existence of certain script, language or feature in a given font.

```lua
local function check_script(id, script)
  local s = string.lower(script)
  if id and id > 0 then
    local otfdata = fonts.identifiers[id].shared.otfdata
    if otfdata then
      local features = otfdata.luatemt.features
      for i, _ in pairs(features) do
        for j, _ in pairs(features[i]) do
          if features[i][j][s] then
            fontspec.log("script '%s' exists in font '%s'", script, fonts.identifiers[id]. fullname)
            return true
          end
        end
      end
    end
  end
end
```

```lua
local function check_language(id, language, script)
  local s = string.lower(script)
  local l = string.lower(language)
  if id and id > 0 then
    local otfdata = fonts.identifiers[id].shared.otfdata
    if otfdata then
      local features = otfdata.luatemt.features
      for i, _ in pairs(features) do
        for j, _ in pairs(features[i]) do
          if features[i][j][s] then
            fontspec.log("script '%s' exists in font '%s'", script, fonts.identifiers[id]. fullname)
            return true
          end
        end
      end
    end
  end
end
```
if otfdata then
  local features = otfdata.luatemx.features
  for i, _ in pairs(features) do
    for j, _ in pairs(features[i]) do
      if features[i][j][s] and features[i][j][s][l] then
        fontspec.log("language '%s' for script '%s' exists in font '%s'",
                      language, script, fonts.identifiers[id].fullname)
        return true
      end
    end
  end
end

local function check_feature(id, feature, language, script)
  local s = string.lower(script)
  local l = string.lower(language)
  local f = string.lower(feature:gsub("^[+-]", "))
  if id and id > 0 then
    local otfdata = fonts.identifiers[id].shared.otfdata
    if otfdata then
      local features = otfdata.luatemx.features
      for i, _ in pairs(features) do
        if features[i][f] and features[i][f][s] then
          if features[i][f][s][l] == true then
            fontspec.log("feature '%s' for language '%s' and script '%s' exists in font '%s'",
                          feature, language, script, fonts.identifiers[id].fullname)
            return true
          end
        end
      end
    end
  end
end

The following are the function that get called from \TeX end.

local function tempswatrue() fontspec.sprint([[\tempswatrue]]) end
local function tempswafalse() fontspec.sprint([[\tempswafalse]]) end

function fontspec.check_ot_script(fnt, script)
  if check_script(font.id(fnt), script) then
    tempswatrue()
  else
    tempswafalse()
  end
end

function fontspec.check_ot_lang(fnt, lang, script)
  if check_language(font.id(fnt), lang, script) then
    tempswatrue()
  else
    tempswafalse()
  end
end
function fontspec.check_ot_feat(fnt, feat, lang, script)
  for _, f in ipairs {"+trep", "+tlig", "+anum"} do
    if feat == f then
      tempswatrue()
      return
    end
  end
  if check_feature(font.id(fnt), feat, lang, script) then
    tempswatrue()
  else
    tempswafalse()
  end
end

function fontspec.mathfontdimen(fnt, str)
  local mathdimens = fonts.identifiers[font.id(fnt)].MathConstants
  if mathdimens then
    local m = mathdimens[str]
    if m then
      fontspec.sprint(mathdimens[str])
      fontspec.sprint("sp")
    else
      fontspec.sprint("/zero.noslashpt")
    end
  else
    fontspec.sprint("/zero.noslashpt")
  end
end

Here we patch fonts tfm table to emulate \TeX's \fontdimen8 which stores
the caps-height of the font. (Cf. \fontdimen5 which stores the x-height.)
Falls back to measuring the glyph if the font doesn't contain the necessary
information. This needs to be extended for fonts that don't contain an 'X'.

local function set_capheight(fontdata)
  local capheight
  local units = fontdata.units
  local size = fontdata.size
  local otfdata = fontdata.shared.otfdata

  if otfdata.pfminfo.os2_capheight > /zero.noslash then
    capheight = otfdata.pfminfo.os2_capheight / units * size
  else
    if fontdata.characters[string.byte("X")] then
      capheight = fontdata.characters[string.byte("X")].height
    else
      capheight = otfdata.metadata.ascent / units * size
    end
  end
  fontdata.parameters[8] = capheight
end

luatexbase.add_to_callback("luaotfload.patch_font", set_capheight, "fontspec.set_capheight")
138 (∋lua)
Part IX
fontspec-patches.sty

1 ⟨∗patches⟩
2 \ExplSyntaxOn

25.11 Unicode footnote symbols
3 \RequirePackage{fixltx2e}[2006/03/24]

25.12 Emph
4 \em\emph
5 \emshape\eminnershape
Redefinition of {\em ...} and \emph{...} to use \nfss info to detect when the inner shape should be used.
6 \DeclareRobustCommand\em{
7 \@nomath\em
8 \str_if_eq:xxTF \f@shape \itdefault \eminnershape
9 {10 \str_if_eq:xxTF \f@shape \sldefault \eminnershape \emshape
11}
12 \DeclareTextFontCommand{\emph}{\em}
13 \cs_set_eq:NN \emshape \itshape
14 \cs_set_eq:NN \eminnershape \upshape

25.13 \-
- This macro is courtesy of Frank Mittelbach and the \LaTeX{}2ε source code.
14 \DeclareRobustCommand\-{\%
15 \discretionary{%16 \char\ifnum\hyphenchar\font<\z@
17 \else
18 \hyphenchar\font
19 \fi\}
20 \def\xlx@defaulthyphenchar{\-}
21}

25.14 Verbatims
Many thanks to Apostolos Syropoulos for discovering this problem and writing the redefinition of \LaTeX{}’s verbatim environment and \verb* command.

\fontspec_visible_space: Print u+2434: OPEN BOX, which is used to visibly display a space character.
If the current font doesn't have \textvisiblespace, use Latin Modern Mono instead.

-helper macro to turn spaces (\textvisiblespace) active and print visible space instead.

\verb R redefine \verb to use \fontspec_print_visible_spaces:

\verb With the \verbatim package.

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

This is for vanilla \LaTeX. 

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim

\verbatim
listings and fancvrb make things nice and easy:

\cs_set:Npn \fontspec_patch_moreverb: { 
  \ifpackageloaded{moreverb}{
    \cs_set:cpn {listingcont*} {
      \cs_set:Npn \verbatim@processline {
        \helisting@line \global\advance\listing@line\c_one
        \the\verbatim@line\par
      }
    }
  }
  \verbatim \fontspec_print_visible_spaces: \verbatim@start
}

\cs_set:Npn \fontspec_patch_fancyvrb: { 
  \ifpackageloaded{fancyvrb}{
    \cs_set_eq:NN \FancyVerbSpace \fontspec_visible_space:
  }
}

\cs_set:Npn \fontspec_patch_listings: { 
  \ifpackageloaded{listings}{
    \cs_set_eq:NN \lst@visiblespace \fontspec_visible_space:
  }
}

\newcommand{\oldstylenums}[1]{
  \addfontfeature{Numbers=OldStyle} #1
}
\newcommand{\liningnums}[1]{
  \addfontfeature{Numbers=Lining} #1
}

25.15 \oldstylenums
\liningnums

This command obviously needs a redefinition. And we may as well provide the reverse command.
\RenewDocumentCommand{\oldstylenums}{m}{
  \addfontfeature{Numbers=OldStyle} #1
}
\NewDocumentCommand{\liningnums}{m}{
  \addfontfeature{Numbers=Lining} #1
}

(/patches)
Part X

\texttt{fontspec.cfg}

1 \texttt{\langle cfg \rangle}

2

3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

4 \texttt{\%\% FOR BACKWARDS COMPATIBILITY WITH PREVIOUS VERSIONS \%\%}

5

6 \texttt{\% Please note that most of the entries here from fontspec v1.x are}

7 \texttt{\% no longer present. Please advise of any serious problems this causes.}

8

9 \texttt{\textbackslash aliasfontfeatureoption\{Ligatures\}\{Historic\}\{Historical\}}

10 \texttt{\textbackslash let\textbackslash newfontinstance\newfontfamily}

11

12 \texttt{\textbackslash /cfg\rangle}
Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\- ........................................ 14, 1037
@@sverb .................................. 45, 47
@ifpackageloaded ................. 55, 66, 77, 82, 2287, 2293, 2300–2302, 2378, 2380–2388, 2390–2392, 2394–2401
@ifstar ..................................... 45
@makeother .................................. 43
@noligs ..................................... 44
@nomath .................................... 5
@onlypreamble .......................... 311–314
@sverb ..................................... 47
@sxverbatim ............................. 61
@tempa ..................................... 2316, 2317
@tempswafalse .............. 78, 1194, 1221, 1252
@tempswatrue ......... 77, 1198, 1225, 1257
verb ........................................ 45
verbatim .......................... 57, 61, 72
8, 10, 66, 74, 75, 126, 162, 179, 188, 194, 195, 217, 222, 228–230, 944, 955, 958
\_fontspec_parse_wordspace:w ........ 1503, 1507
_int_mult_truncate:Nn ................ 51, 1606

A
acute .................................. 2305
addfontfeature .......... 88, 92, 117, 350
addfontfeatures ............ 334
advance .................................. 69
aliasfontfeature .............. 376, 1291, 1601, 1878
aliasfontfeatureoption ...... 9, 376
\AtBeginDocument .................. 48, 2407

B
bar ....................................... 2309
bfdefault ..................... 763, 766, 808, 817, 821, 948, 949, 952, 953, 2362, 2365, 2367, 2368, 2371, 2374, 2375
\group .................................... 42
bool_if:nF ..................... 760, 771, 860, 878, 1067, 1295, 1439, 1502, 1583, 1595, 1609, 2321, 2326
bool_if:nF ....................... 788
bool_if:NT .......................... 42
.. 704, 846, 965, 985, 1052, 1077, 1664, 1670, 1710, 1731, 2402, 2408

bool_if:nT .......................... 933, 1669
bool_if:NF ..................... 465, 479, 488, 505, 520, 533, 548, 563, 577, 677, 843, 1049, 1074, 1660, 1972, 2294
bool_if:nTF ....................... 1180, 2263
bool_new:N ................... 21–34, 1277
bool_set_true:N ............. 236, 238, 675, 833, 835, 837, 840, 855, 1028, 1280–1282, 1350, 1363, 1393, 1396, 2288, 2295, 2300–2302
bool_until_do:nn .......... 1195, 1222, 1253
breve .................................. 2310
\c_colon_str ..................... 43, 1883, 1895
\c_empty_tl ..................... 1168, 1174, 1175
\c_keys_code_root_tl .......... 404, 405
\c_minus_one .................. 1547
\c_one .................................. 69
\c_zero .................................. 836, 1561
\char .................................. 16, 24
\char_set_catcode_active:n ........ 34, 36
\check .................................. 2311
\clist_clear:N .................. 1021
\clist_if_empty:NTF .......... 895
\clist_map_break: ............. 675
\clist_map_inline:N .................. 671, 908
\clist_map_inline:nn ...... 1861, 1874, 1892
\clist_put_right:N .............. 1713, 1734
\clist_put_right:Nx 1721, 1723, 1741, 1743
\colon .................................... 1788
\clist_map_inline:nn ...... 2317, 2318
\color@ .................................. 1572
\convertcolorspec ............. 1574
\cs_generate_variant:Nn .................. 46–50, 728, 729, 1065, 1088, 1123, 1178, 1407
\cs_gset:Npn ..................... 35, 1283
\cs_if_exist:cF .............. 685
\cs_if_exist:CF .................. 456, 687, 700, 1572
\cs_if_exist_p:c .................. 2266
\cs_if_free:NT .................. 598

C
\appendix .................................. 16
\textbf{.................. 16
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Gamma</td>
<td>2337</td>
</tr>
<tr>
<td>\global</td>
<td>69, 604</td>
</tr>
<tr>
<td>grave</td>
<td>2306</td>
</tr>
<tr>
<td>\group_begin:</td>
<td>33, 57, 336, 872, 1476, 2315</td>
</tr>
<tr>
<td>\group_end:</td>
<td>39, 343, 658, 892, 1485, 2320</td>
</tr>
<tr>
<td>H</td>
<td>2312</td>
</tr>
<tr>
<td>\hbox</td>
<td>41</td>
</tr>
<tr>
<td>\hyphenchar</td>
<td>16, 19, 1547, 1557, 1561</td>
</tr>
<tr>
<td>I</td>
<td>1204, 1210, 1231, 1239, 1263, 1274</td>
</tr>
<tr>
<td>\if@tempswa</td>
<td>1231</td>
</tr>
<tr>
<td>\ifcase</td>
<td>832</td>
</tr>
<tr>
<td>\ifcsname</td>
<td>335</td>
</tr>
<tr>
<td>\ifdimexpr</td>
<td>41</td>
</tr>
<tr>
<td>\ifnum</td>
<td>16, 836, 1197, 1224, 1255</td>
</tr>
<tr>
<td>\ifx</td>
<td>610, 1174, 1175, 2317, 2379, 2389, 2393</td>
</tr>
<tr>
<td>\ignorespaces</td>
<td>280, 348</td>
</tr>
<tr>
<td>\InputIfFileExists</td>
<td>2409</td>
</tr>
<tr>
<td>\int_compare:nT</td>
<td>1615</td>
</tr>
<tr>
<td>\int_compare:nTF</td>
<td>1145, 1304, 1577, 1580</td>
</tr>
<tr>
<td>\int_compare:n:N</td>
<td>1195, 1222, 1253</td>
</tr>
<tr>
<td>\int_eval:n</td>
<td>47, 53, 419, 442, 1171, 1192, 1217, 1226, 1245, 1258, 1605, 2224, 2229</td>
</tr>
<tr>
<td>\int_gincr:c</td>
<td>688</td>
</tr>
<tr>
<td>\int_if_even:nF</td>
<td>1149</td>
</tr>
<tr>
<td>\int_new:c</td>
<td>1201, 1228, 1260</td>
</tr>
<tr>
<td>\int_new:N</td>
<td>690</td>
</tr>
<tr>
<td>\int_new:N</td>
<td>35–37, 1602</td>
</tr>
<tr>
<td>\int_set:N:n</td>
<td>47, 53, 419, 442, 1171, 1192, 1217, 1226, 1245, 1258, 1605, 2224, 2229</td>
</tr>
<tr>
<td>\int_set:Nn</td>
<td>489, 490, 535</td>
</tr>
<tr>
<td>\int_to_hexadecimal:n</td>
<td>1616</td>
</tr>
<tr>
<td>\int_use:c</td>
<td>696</td>
</tr>
<tr>
<td>\int_zero:N</td>
<td>1193, 1220, 1251, 2245</td>
</tr>
<tr>
<td>\itdefault</td>
<td>6, 775, 808, 881, 899, 934, 935, 940, 950, 952, 2273, 2281, 2361, 2368, 2372</td>
</tr>
<tr>
<td>\itshape</td>
<td>12, 2271</td>
</tr>
<tr>
<td>\keys_if_choice_exist:nnT</td>
<td>364, 371</td>
</tr>
<tr>
<td>\keys_if_exist:nnF</td>
<td>362, 369</td>
</tr>
<tr>
<td>\keys_if_exist:nnTF</td>
<td>378, 384, 390</td>
</tr>
<tr>
<td>\keys_set:nn</td>
<td>49, 381, 387, 395, 423, 445, 662, 1287, 1296, 1331, 1337, 1635, 2234</td>
</tr>
<tr>
<td>\keys_set:nx</td>
<td>712, 713, 723, 724, 995</td>
</tr>
<tr>
<td>\keys_set_known:nnN</td>
<td>50</td>
</tr>
<tr>
<td>\keys_set_known:nnN</td>
<td>663, 666, 913</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>_fontspec_atsui_bool</td>
<td>26, 465, 828, 835, 843, 1052, 1077</td>
</tr>
<tr>
<td>_fontspec_defined_shapes_t1</td>
<td>197, 943, 1034</td>
</tr>
<tr>
<td>_fontspec_extension_t1</td>
<td>680, 1294, 1300</td>
</tr>
<tr>
<td>_fontspec_extensions_clist</td>
<td>451, 452, 671</td>
</tr>
<tr>
<td>_fontspec_external_bool</td>
<td>860, 1277, 1282, 1295</td>
</tr>
<tr>
<td>_fontspec_fake_embolden_t1</td>
<td>1005, 1719, 1722, 1733</td>
</tr>
<tr>
<td>_fontspec_fake_slant_t1</td>
<td>1004, 1712, 1740, 1742</td>
</tr>
<tr>
<td>_fontspec_family_t1</td>
<td>193, 323, 344, 389, 595, 649, 699, 700, 732–735, 739–742, 744–746, 750–753, 930, 939, 940, 2417</td>
</tr>
<tr>
<td>_fontspec_feature_string_t1</td>
<td>1096, 1162</td>
</tr>
<tr>
<td>_fontspec_firsttime_bool</td>
<td>21, 646, 704, 985, 1028, 1067, 1439, 1502, 1583, 1595, 1609, 1670, 1710, 1731</td>
</tr>
<tr>
<td>_fontspec_font</td>
<td>464, 477, 486, 503, 518, 531, 546, 561, 575, 594, 639, 640, 642, 643, 832, 836, 874, 875, 886, 1141, 1145, 1147, 1151, 1154, 1192, 1197, 1218, 1224, 1246, 1255, 1479, 1553, 2418</td>
</tr>
<tr>
<td>_fontspec_fontfeat_bf_clist</td>
<td>763, 766, 1011, 1414, 1734</td>
</tr>
<tr>
<td>_fontspec_fontfeat_bfit_clist</td>
<td>808, 1013, 1424, 1721, 1723, 1741, 1743</td>
</tr>
<tr>
<td>_fontspec_fontfeat_bfs_clist</td>
<td>817, 821, 1015, 1434</td>
</tr>
<tr>
<td>_fontspec_fontfeat_clist</td>
<td>667, 995</td>
</tr>
<tr>
<td>_fontspec_fontfeat_it_clist</td>
<td>775, 1012, 1419, 1713</td>
</tr>
<tr>
<td>_fontspec_fontfeat_sc_clist</td>
<td>882, 890, 1016, 1440</td>
</tr>
</tbody>
</table>