The fontspec package
Font selection for XeLaTeX and LuaLaTeX

Will Robertson and Khaled Hosny
http://wspr.io/fontspec/

2017/11/05 v2.6f

Contents

1 Loading 3

2 Declaration of variables 3

3 Error/warning/info messages 6
- 3.1 Errors . . . 7
- 3.2 Warnings . . . 8
- 3.3 Info messages . . . 10

4 Opening code 11
- 4.1 Package options . . . 11
- 4.2 Encodings . . . 12
- 4.3 Generic functions . . . 13
- 4.4 expl3 variants . . . 13

5 expl3 interface for primitive font loading 14

6 User commands 15

7 User command internals 17
- 7.1 Font selection . . . 17
- 7.2 Font feature selection . . . 21
- 7.3 Defining new font features . . . 22

8 Programmer's interface 25

9 Internals 31
- 9.1 The main function for setting fonts . . . 31
- 9.2 Setting font shapes in a family . . . 39
- 9.3 Initialisation . . . 48
- 9.4 Miscellaneous . . . 49

10 OpenType definitions code 49

10.1 Adding features when loading fonts . . . 51
10.2 OpenType feature information . . . 54

11 Graphite/AAT code 56

12 Font loading (keyval) definitions 58
12.1 OpenType feature definitions . . . 72
12.2 Regular key=val / tag definitions . . . 72
12.3 OpenType features that need numbering . . . 77
12.4 Script and Language . . . 78
12.5 Backwards compatibility . . . 80
12.6 Font script definitions . . . 81
12.7 Font language definitions . . . 84
12.8 AAT feature definitions . . . 91

13 Extended font encodings 95

14 Selecting maths fonts 97

15 Closing code 101
15.1 Finishing up . . . 101

16 Changes to the NFSS 101
16.1 Italic small caps and so on . . . 101
16.2 Emphasis . . . 103
16.3 Strong emphasis . . . 104

17 Patching code 106
17.1 \- . . . 106
17.2 Verbatims . . . 106
17.3 \oldstylenums . . . 108

Index 109

1 Loading

The expl3 module is `fontspec`.

```
⟨@@=fontspec⟩
```

Check engine and load specific modules. For LuaTEX, load luaotfload.

```
⟨*load⟩
\sys_if_engine_luatex:T
  { \RequirePackage{luaotfload}
    \directlua{require("fontspec")}
    \RequirePackageWithOptions{fontspec-luatex} \endinput }
\sys_if_engine_xetex:T
  { \RequirePackageWithOptions{fontspec-xetex}  \endinput }
```

If not one of the above, error:

```
\msg_new:nnn {fontspec} {cannot-use-pdftex}
 {
  The~ fontspec~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\\\
  You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~ "xelatex"~ or~ "lualatex" instea
 }
\msg_fatal:nn {fontspec} {cannot-use-pdftex}
\endinput
⟨/load⟩
```

2 Declaration of variables

```
⟨*fontspec⟩
```

Booleans

firsttime 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 occuring per-shape this no longer needs to happen; this is indicated by the 'firsttime' conditional.

```
\bool_new:N \l_@@_firsttime_bool
\bool_new:N \l_@@_nobf_bool
\bool_new:N \l_@@_noit_bool
\bool_new:N \l_@@_nosc_bool
\bool_new:N \l_@@_check_bool
\bool_new:N \l_@@_tfm_bool
\bool_new:N \l_@@_atsui_bool
\bool_new:N \l_@@_ot_bool
\bool_new:N \l_@@_mm_bool
\bool_new:N \l_@@_graphite_bool
\bool_new:N \l_@@_fontcfg_bool
\bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

```
\bool_new:N \g_@@_math_euler_bool
\bool_new:N \g_@@_math_lucida_bool
\bool_new:N \g_@@_pkg_euler_loaded_bool
```

For package options:

```
\bool_new:N \g_@@_cfg_bool
\bool_new:N \g_@@_math_bool
\bool_new:N \g_@@_euenc_bool
\bool_new:N \l_@@_tmpa_bool
\bool_new:N \l_@@_disable_defaults_bool
\bool_new:N \l_@@_alias_bool
\bool_new:N \l_@@_external_bool
\bool_new:N \l_@@_never_check_bool
\bool_new:N \l_@@_defining_encoding_bool
\bool_new:N \l_@@_script_exist_bool
\bool_new:N \g_@@_em_normalise_slant_bool
```

Counters

```
\int_new:N \l_@@_script_int
\int_new:N \l_@@_language_int
\int_new:N \l_@@_strnum_int
\int_new:N \l_@@_tmp_int
\int_new:N \l_@@_em_int
\int_new:N \l_@@_emdef_int
\int_new:N \l_@@_strong_int
\int_new:N \l_@@_strongdef_int
```

Floating point

```
\fp_new:N \l_@@_tmpa_fp
\fp_new:N \l_@@_tmpb_fp
```

Dimensions

```
\dim_new:N \l_@@_tmpa_dim
\dim_new:N \l_@@_tmpb_dim
\dim_new:N \l_@@_tmpc_dim
\seq_new:N \g_@@_bf_series_seq
```

Comma lists

```
\clist_new:N \g_@@_default_fontopts_clist
\clist_new:N \g_@@_all_keyval_modules_clist
\clist_new:N \l_@@_sizefeat_clist
\clist_set:Nn \l_@@_sizefeat_clist {Size={-}}
\clist_new:N \l_@@_extensions_clist
\clist_new:N \l_@@_fontopts_clist
\clist_new:N \l_@@_family_fontopts_clist
\clist_new:N \l_@@_all_features_clist
\clist_new:N \l_@@_leftover_clist
\clist_new:N \l_@@_keys_leftover_clist
\clist_new:N \l_@@_sizing_leftover_clist
\clist_new:N \l_@@_fontfeat_clist
\clist_new:N \l_@@_fontfeat_curr_clist
```

```
\tl_new:N \l_@@_fontfeat_up_clist
\tl_new:N \l_@@_fontfeat_bf_clist
\tl_new:N \l_@@_fontfeat_it_clist
\tl_new:N \l_@@_fontfeat_bfit_clist
\tl_new:N \l_@@_fontfeat_sl_clist
\tl_new:N \l_@@_fontfeat_bfsl_clist
\tl_new:N \l_@@_fontfeat_sc_clist
```

Property lists

```
\prop_new:N \g_@@_fontopts_prop
\prop_new:N \l_@@_nfss_prop
\prop_new:N \l_@@_nfssfont_prop
\prop_new:N \g_@@_OT_features_prop
\prop_new:N \g_@@_all_opentype_feature_names_prop
\prop_new:N \g_@@_em_prop
\prop_new:N \g_@@_strong_prop
```

Token lists

```
\tl_new:N \l_fontspec_family_tl
\tl_new:N \g_fontspec_encoding_tl
\tl_new:N \l_fontspec_renderer_tl
\tl_new:N \l_fontspec_fontname_tl
\tl_new:N \l_fontspec_defined_shapes_tl
```

```
\tl_new:N \UTFencname
\tl_new:N \cyrillicencoding
\tl_new:N \latinencoding
```

```
\tl_new:N \g_@@_single_feat_tl
\tl_new:N \l_@@_tmp_tl
\tl_new:N \l_@@_size_tl
\tl_new:N \l_@@_sizedfont_tl
\tl_new:N \l_@@_nfss_tl
\tl_new:N \l_@@_nfss_sc_tl
\tl_new:N \l_@@_this_font_tl
\tl_new:N \l_@@_scale_tl
\tl_new:N \l_@@_opacity_tl
\tl_new:N \l_@@_hexcol_tl
\tl_new:N \l_@@_fontid_tl
\tl_new:N \l_@@_extension_tl
\tl_new:N \l_@@_ext_filename_tl
\tl_new:N \l_@@_font_path_tl
\tl_new:N \l_@@_basename_tl
\tl_new:N \l_@@_curr_fontname_tl
\tl_new:N \l_@@_saved_fontname_tl
\tl_new:N \l_@@_optical_size_tl
\tl_new:N \l_@@_ttc_index_tl
\tl_new:N \l_@@_nfss_enc_tl
\tl_new:N \g_@@_curr_series_tl
\tl_new:N \l_@@_options_tl
\tl_new:N \l_@@_fontname_tl
```

```
\tl_new:N \l_@@_rawfeatures_sclist
\tl_new:N \l_@@_pre_feat_sclist

\tl_new:N \g_@@_rmfamily_family
\tl_new:N \g_@@_sffamily_family
\tl_new:N \g_@@_ttfamily_family

\tl_new:N \g_@@_mathrm_tl
\tl_new:N \g_@@_bfmathrm_tl
\tl_new:N \g_@@_mathsf_tl
\tl_new:N \g_@@_mathtt_tl

\tl_new:N \l_@@_family_label_tl
\tl_new:N \l_@@_fake_slant_tl
\tl_new:N \l_@@_fake_embolden_tl

\tl_new:N \l_@@_fontname_up_tl
\tl_new:N \l_@@_fontname_bf_tl
\tl_new:N \l_@@_fontname_it_tl
\tl_new:N \l_@@_fontname_bfit_tl
\tl_new:N \l_@@_fontname_sl_tl
\tl_new:N \l_@@_fontname_bfsl_tl
\tl_new:N \l_@@_fontname_sc_tl

\tl_new:N \l_@@_script_name_tl
\tl_new:N \l_fontspec_script_tl
\tl_new:N \l_@@_lang_name_tl
\tl_new:N \l_fontspec_lang_tl

\tl_new:N  \l_@@_mapping_tl
\tl_new:N  \g_@@_hexcol_tl
\tl_new:N  \g_@@_opacity_tl
\tl_set:Nn \g_@@_hexcol_tl {000000}
\tl_set:Nn \g_@@_opacity_tl {FF~}
\tl_new:N  \l_@@_punctspace_adjust_tl
\tl_new:N  \l_@@_wordspace_adjust_tl
\tl_new:N  \l_@@_postadjust_tl
\tl_new:N  \g_@@_postadjust_tl
\tl_set:Nn \g_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }
```

⟨/fontspec⟩

3 Error/warning/info messages

⟨*fontspec⟩

Shorthands for messages:

```
\cs_new:Npn \@@_error:n     { \msg_error:nn     {fontspec} }
\cs_new:Npn \@@_error:nn    { \msg_error:nnn    {fontspec} }
\cs_new:Npn \@@_error:nx    { \msg_error:nnx    {fontspec} }
\cs_new:Npn \@@_warning:n   { \msg_warning:nn   {fontspec} }
\cs_new:Npn \@@_warning:nx  { \msg_warning:nnx  {fontspec} }
\cs_new:Npn \@@_warning:nxx { \msg_warning:nnxx {fontspec} }
\cs_new:Npn \@@_info:n      { \msg_info:nn      {fontspec} }
\cs_new:Npn \@@_info:nx     { \msg_info:nnx     {fontspec} }
```

```
\cs_new:Npn \@@_info:nxx    { \msg_info:nnxx    {fontspec} }
\cs_new:Npn \@@_trace:n     { \msg_trace:nn     {fontspec} }
```

Allow messages to be written with spaces acting as normal:

```
\cs_generate_variant:Nn \msg_new:nnn  {nnx}
\cs_generate_variant:Nn \msg_new:nnnn {nnxx}
\cs_new:Nn \@@_msg_new:nnn
  { \msg_new:nnx {#1} {#2} { \tl_trim_spaces:n {#3} } }
\cs_new:Nn \@@_msg_new:nnnn
  { \msg_new:nnxx {#1} {#2} { \tl_trim_spaces:n {#3} } { \tl_trim_spaces:n {#4} } }
\char_set_catcode_space:n {32}
```

3.1 Errors

```
\@@_msg_new:nnn {fontspec} {only-inside-encdef}
 {
  \exp_not:N#1can only be used in the second argument
  to \string\DeclareUnicodeEncoding.
 }
\@@_msg_new:nnn {fontspec} {only-import-tu}
 {
  The "\string\ImportEncoding" command can only take "TU" as an argument at this stage.
 }
\@@_msg_new:nnn {fontspec} {no-size-info}
 {
  Size information must be supplied.\\
  For example, SizeFeatures={Size={8-12},...}.
 }
\@@_msg_new:nnnn {fontspec} {font-not-found}
 {
  The font "#1" cannot be found.
 }
 {
  A font might not be found for many reasons.\\
  Check the spelling, where the font is installed etc. etc.\\\\
  When in doubt, ask someone for help!
 }
\@@_msg_new:nnnn {fontspec} {rename-feature-not-exist}
 {
  The feature #1 doesn't appear to be defined.
 }
 {
  It looks like you're trying to rename a feature that doesn't exist.
 }
\@@_msg_new:nnn {fontspec} {no-glyph}
 {
  '\l_fontspec_fontname_tl' does not contain glyph #1.
 }
\@@_msg_new:nnnn {fontspec} {euler-too-late}
 {
  The euler package must be loaded BEFORE fontspec.
 }
 {
```

```
  fontspec only overwrites euler's attempt to
  define the maths text fonts if fontspec is
  loaded after euler. Type <return> to proceed
  with incorrect \string\mathit, \string\mathbf, etc.
 }
\@@_msg_new:nnnn {fontspec} {no-xcolor}
 {
  Cannot load named colours without the xcolor package.
 }
 {
  Sorry, I can't do anything to help. Instead of loading
  the color package, use xcolor instead.
 }
\@@_msg_new:nnnn {fontspec} {unknown-color-model}
 {
  Error loading colour `#1'; unknown colour model.
 }
 {
  Sorry, I can't do anything to help. Please report this error
  to my developer with a minimal example that causes the problem.
 }
\@@_msg_new:nnnn {fontspec} {not-in-addfontfeatures}
 {
  The "#1" font feature cannot be used in \string\addfontfeatures.
 }
 {
  This is due to how TeX loads fonts; such settings
  are global so adding them mid-document within a group causes
  confusion. You'll need to define multiple font families to achieve
  what you want.
 }
```

3.2 Warnings

```
\@@_msg_new:nnn {fontspec} {tu-clash}
 {
  I have found the tuenc.def encoding definition file but the TU encoding is not
  defined by the LaTeX2e kernel; attempting to correct but you really should update
  to the latest version of LaTeX2e.
 }
\@@_msg_new:nnn {fontspec} {tu-missing}
 {
  The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
 }
\@@_msg_new:nnn {fontspec} {addfontfeatures-ignored}
 {
  \string\addfontfeature (s) ignored \msg_line_context:;
  it cannot be used with a font that wasn't selected by a fontspec command.\\
  \\
  The current font is "\use:c{font@name}".\\
  \int_compare:nTF { \clist_count:n {#1} = 1 }
    { The requested feature is "#1". }
```

```
    { The requested features are "#1". }
 }
\@@_msg_new:nnn {fontspec} {feature-option-overwrite}
 {
  Option '#2' of font feature '#1' overwritten.
 }
\@@_msg_new:nnn {fontspec} {script-not-exist-latn}
 {
  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 OpenType 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_@@_script_name_tl' and language '\l_@@_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 script '\l_@@_script_name_tl'.\\
  'Default' language used instead.
 }
\@@_msg_new:nnn {fontspec} {only-xetex-feature}
 {
  Ignored XeTeX only feature: '#1'.
 }
```

```
\@@_msg_new:nnn {fontspec} {only-luatex-feature}
 {
  Ignored LuaTeX only feature: '#1'.
 }
\@@_msg_new:nnn {fontspec} {no-mapping}
 {
  Input mapping not (yet?) supported in LuaTeX.
 }
\@@_msg_new:nnn {fontspec} {no-mapping-ligtex}
 {
  Input mapping not (yet?) supported in LuaTeX.\\
  Use "Ligatures=TeX" instead of "Mapping=tex-text".
 }
\@@_msg_new:nnn {fontspec} {cm-default-obsolete}
 {
  The "cm-default" package option is obsolete.
 }
\@@_msg_new:nnn {fontspec} {fakebold-only-xetex}
 {
  The "FakeBold" and "AutoFakeBold" options are only available with XeLaTeX.\\
  Option ignored.
 }
\@@_msg_new:nnn {fontspec} {font-index-needs-ttc}
 {
  The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
  Feature ignored.
 }
\@@_msg_new:nnn {fontspec} {feat-cannot-remove}
 {
  The "#1" feature cannot be deactivated. Request ignored.
 }
```

3.3 Info messages

```
\@@_msg_new:nnn {fontspec} {defining-font}
 {
  Font family '\l_fontspec_family_tl' created for font '#2'
  with options [\l_@@_all_features_clist].\\
  \\
  This font family consists of the following NFSS series/shapes:\\
  \l_fontspec_defined_shapes_tl
 }
\@@_msg_new:nnn {fontspec} {no-font-shape}
 {
  Could not resolve font "#1" (it probably doesn't exist).
 }
\@@_msg_new:nnn {fontspec} {set-scale}
 {
  \l_fontspec_fontname_tl\space scale = \l_@@_scale_tl.
 }
\@@_msg_new:nnn {fontspec} {setup-math}
 {
```

```
  Adjusting the maths setup (use [no-math] to avoid this).
 }
\@@_msg_new:nnn {fontspec} {no-scripts}
 {
  Font "\l_fontspec_fontname_tl" 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
 }
```

Reset 'space' behaviour:

```
\char_set_catcode_ignore:n {32}
```

⟨/fontspec⟩

4 Opening code

4.1 Package options

```
\DeclareOption{cm-default}
 { \@@_warning:n {cm-default-obsolete} }
\DeclareOption{math}{\bool_set_true:N \g_@@_math_bool}
\DeclareOption{no-math}{\bool_set_false:N \g_@@_math_bool}
\DeclareOption{config}{\bool_set_true:N \g_@@_cfg_bool}
\DeclareOption{no-config}{\bool_set_false:N \g_@@_cfg_bool}
\DeclareOption{euenc}{\bool_set_true:N  \g_@@_euenc_bool}
\DeclareOption{tuenc}{\bool_set_false:N \g_@@_euenc_bool}
\DeclareOption{quiet}
 {
  \msg_redirect_module:nnn { fontspec } { warning } { info }
  \msg_redirect_module:nnn { fontspec } { info } { none }
 }
\DeclareOption{silent}
 {
  \msg_redirect_module:nnn { fontspec } { warning } { none }
  \msg_redirect_module:nnn { fontspec } { info } { none }
 }
\ExecuteOptions{config,math,tuenc}
```

```
\ProcessOptions*
```

4.2 Encodings

Soon to be the default, with a just-in-case check:

```
\bool_if:NF \g_@@_euenc_bool
  {
    \file_if_exist:nTF {tuenc.def}
      {
        \cs_if_exist:cF {T@TU}
          {
            \@@_warning:n {tu-clash}
            \DeclareFontEncoding{TU}{}{}
            \DeclareFontSubstitution{TU}{lmr}{m}{n}
          }
      }
      {
        \@@_warning:n {tu-missing}
        \bool_set_true:N \g_@@_euenc_bool
      }
  }
\bool_if:NTF \g_@@_euenc_bool
  {
⟨xetex⟩    \tl_set:Nn \g_fontspec_encoding_tl {EU1}
⟨luatex⟩   \tl_set:Nn \g_fontspec_encoding_tl {EU2}
  }
  { \tl_set:Nn \g_fontspec_encoding_tl { TU } }
\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 if needed
```

To overcome the encoding changing the current font size, but only if a class has been loaded first:

```
\tl_if_in:NnT \@filelist {.cls} { \normalsize }
```

Dealing with a couple of the problems introduced by babel:

```
\tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
\tl_set_eq:NN \latinencoding    \g_fontspec_encoding_tl
\AtBeginDocument
 {
  \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
  \tl_set_eq:NN \latinencoding    \g_fontspec_encoding_tl
 }
```

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

```
\bool_if:NT \g_@@_euenc_bool
  {
⟨luatex⟩   \cs_set_eq:NN \fontspec_tmp: \XeTeXpicfile
⟨luatex⟩   \cs_set:Npn \XeTeXpicfile {}
```

```
    \RequirePackage{xunicode}
⟨luatex⟩    \cs_set_eq:NN \XeTeXpicfile \fontspec_tmp:
  }
```

4.3 Generic functions

\FontspecSetCheckBoolTrue
\FontspecSetCheckBoolFalse

These strange set functions are to simplify returning code from LuaTeX:

```
\cs_new:Npn \FontspecSetCheckBoolTrue  { \bool_set_true:N  \l_@@_check_bool }
\cs_new:Npn \FontspecSetCheckBoolFalse { \bool_set_false:N \l_@@_check_bool }
```

(*End definition for* `\FontspecSetCheckBoolTrue` *and* `\FontspecSetCheckBoolFalse`*. These functions are documented on page* ??.)

\@@_keys_set_known:nnN

```
\cs_new:Nn \@@_keys_set_known:nnN
  {
⟨debug⟩  \typeout{:::: Keys~set:~{#1}~{#2} }
    \keys_set_known:nnN {#1} {#2} #3
⟨debug⟩  \typeout{:::: Leftover:~{#3} }
  }
\cs_generate_variant:Nn \@@_keys_set_known:nnN {nx}
```

(*End definition for* `\@@_keys_set_known:nnN`*. This function is documented on page* ??.)

\@@_int_mult_truncate:Nn

Missing in expl3, IMO.

```
\cs_new:Nn \@@_int_mult_truncate:Nn
  {
    \int_set:Nn #1 { \__dim_eval:w #2 #1 \__dim_eval_end: }
  }
```

(*End definition for* `\@@_int_mult_truncate:Nn`*. This function is documented on page* ??.)

4.4 expl3 variants

```
\cs_generate_variant:Nn \int_set:Nn {Nv}
\cs_generate_variant:Nn \keys_set:nn {nx}
\cs_generate_variant:Nn \keys_set_known:nnN {nx}
\cs_generate_variant:Nn \prop_put:Nnn {Nxx}
\cs_generate_variant:Nn \prop_put:Nnn {NxV}
\cs_generate_variant:Nn \prop_gput_if_new:Nnn  {NxV}
\cs_generate_variant:Nn \prop_gput:Nnn  {Nxn}
\cs_generate_variant:Nn \prop_get:NnNT  {NxN}
\cs_generate_variant:Nn \prop_get:NnNTF {NxN}
\cs_generate_variant:Nn \str_if_eq:nnTF {nv}
\cs_generate_variant:Nn \tl_if_empty:nTF {x}
\cs_generate_variant:Nn \tl_if_empty:nF {x}
\cs_generate_variant:Nn \tl_if_empty:nF {f}
\cs_generate_variant:Nn \tl_if_eq:nnT {ox}
\cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}
```

5 expl3 interface for primitive font loading

\@@_primitive_font_set:Nnn
\@@_primitive_font_gset:Nnn

```
\cs_set:Npn \@@_primitive_font_set:Nnn #1#2#3
  {
    \font #1 = #2 ~at~ #3 \scan_stop:
  }
\cs_set:Npn \@@_primitive_font_gset:Nnn #1#2#3
  {
    \global \font #1 = #2 ~at~ #3 \scan_stop:
  }
```

(*End definition for* `\@@_primitive_font_set:Nnn` *and* `\@@_primitive_font_gset:Nnn`*. These functions are documented on page* ??.)

\@@_font_suppress_not_found_error:

```
\cs_set:Npn \@@_font_suppress_not_found_error:
  {
    \int_set_eq:NN \xetex_suppressfontnotfounderror:D \c_one
  }
```

(*End definition for* `\@@_font_suppress_not_found_error:`*. This function is documented on page* ??.)

\@@_primitive_font_if_null_p:N
\@@_primitive_font_if_null:N*TF*

```
\prg_set_conditional:Nnn \@@_primitive_font_if_null:N {p,TF,T,F}
  {
    \ifx #1 \nullfont
      \prg_return_true:
    \else
      \prg_return_false:
    \fi
  }
```

(*End definition for* `\@@_primitive_font_if_null:NTF`*. This function is documented on page* ??.)

\@@_primitive_font_if_exist:n*TF*

```
\prg_set_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
  {
    \group_begin:
      \@@_font_suppress_not_found_error:
      \@@_primitive_font_set:Nnn \l_@@_primitive_font {#1} {10pt}
      \@@_primitive_font_if_null:NTF \l_@@_primitive_font
        { \group_end: \prg_return_false: }
        { \group_end: \prg_return_true:  }
  }
```

(*End definition for* `\@@_primitive_font_if_exist:nTF`*. This function is documented on page* ??.)

\@@_primitive_font_glyph_if_exist:NnTF

```
\prg_new_conditional:Nnn \@@_primitive_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 definition for* `\@@_primitive_font_glyph_if_exist:NnTF`. *This function is documented on page* ??.)

\@@_primitive_font_set_hyphenchar:Nn

```
\cs_new:Nn \@@_primitive_font_set_hyphenchar:Nn
  {
     \tex_hyphenchar:D #1 = #2 \scan_stop:
  }
```

(*End definition for* `\@@_primitive_font_set_hyphenchar:Nn`. *This function is documented on page* ??.)

6 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 7 on page 17.

```
\NewDocumentCommand \fontspec { O{} m O{} }
  {
    \@@_main_fontspec:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setmainfont { O{} m O{} }
  {
    \@@_main_setmainfont:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setsansfont { O{} m O{} }
  {
    \@@_main_setsansfont:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setmonofont { O{} m O{} }
  {
    \@@_main_setmonofont:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setmathrm { O{} m O{} }
  {
    \@@_main_setmathrm:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setboldmathrm { O{} m O{} }
  {
    \@@_main_setboldmathrm:nn {#1,#3} {#2}
  }
```

```
\NewDocumentCommand \setmathsf { O{} m O{} }
  {
    \@@_main_setmathsf:nn {#1,#3} {#2}
  }
\NewDocumentCommand \setmathtt { O{} m O{} }
  {
    \@@_main_setmathtt:nn {#1,#3} {#2}
  }
```

\setromanfont This is the old name for \setmainfont, retained *ad infinitum* for backwards compatibility. It was deprecated in 2010.

```
\NewDocumentCommand \setromanfont { O{} m O{} }
  {
    \@@_main_setmainfont:nn {#1,#3} {#2}
  }
```

(*End definition for* \setromanfont*. This function is documented on page* ??.)

```
\NewDocumentCommand \newfontfamily { m O{} m O{} }
  {
    \@@_main_newfontfamily:nnn {#1} {#2,#4} {#3}
  }
\NewDocumentCommand \newfontface { m O{} m O{} }
  {
    \@@_main_newfontface:nnn {#1} {#2,#4} {#3}
  }
\NewDocumentCommand \defaultfontfeatures { t+ o m }
  {
    \@@_main_defaultfontfeatures:nnn {#1} {#2} {#3}
  }
\NewDocumentCommand \addfontfeatures {m}
  {
    \@@_main_addfontfeatures:n {#1}
  }
\NewDocumentCommand \addfontfeature  {m}
  {
    \@@_main_addfontfeatures:n {#1}
  }
\NewDocumentCommand \newfontfeature {mm}
  {
    \@@_main_newfontfeature:nn {#1} {#2}
  }
\NewDocumentCommand \newAATfeature {mmmm}
  {
    \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
  }
\NewDocumentCommand \newopentypefeature {mmm}
  {
    \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
  }
```

\newICUfeature Deprecated.

```
\NewDocumentCommand \newICUfeature {mmm}
  {
    \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
  }
```

(*End definition for* `\newICUfeature`. *This function is documented on page* ??.)

```
\NewDocumentCommand \aliasfontfeature {mm}
  {
    \@@_main_aliasfontfeature:nn {#1} {#2}
  }
\NewDocumentCommand \aliasfontfeatureoption {mmm}
  {
    \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
  }
```

\newfontscript Mostly used internally, but also possibly useful for users, to define new OpenType 'scripts', mapping logical names to OpenType script tags.

```
\NewDocumentCommand \newfontscript {mm}
  {
    \fontspec_new_script:nn {#1} {#2}
  }
```

(*End definition for* `\newfontscript`. *This function is documented on page* ??.)

\newfontlanguage Mostly used internally, but also possibly useful for users, to define new OpenType 'languages', mapping logical names to OpenType language tags.

```
\NewDocumentCommand \newfontlanguage {mm}
  {
    \fontspec_new_lang:nn {#1} {#2}
  }
```

(*End definition for* `\newfontlanguage`. *This function is documented on page* ??.)

```
\NewDocumentCommand \DeclareFontsExtensions {m}
  {
    \@@_main_DeclareFontsExtensions:n {#1}
  }
\NewDocumentCommand \IfFontFeatureActiveTF {mmm}
  {
    \@@_main_IfFontFeatureActiveTF:nnn {#1} {#2} {#3}
  }
```

7 User command internals

7.1 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.

```
\cs_set:Nn \@@_main_fontspec:nn
 {
  \fontspec_set_family:Nnn \f@family {#1} {#2}
  \fontencoding { \l_@@_nfss_enc_tl }
  \selectfont
  \ignorespaces
 }
```

(*End definition for* `\fontspec`*. This function is documented on page* ??.)

`\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.

```
\cs_set:Nn \@@_main_setmainfont:nn
 {
  \fontspec_set_family:Nnn \g_@@_rmfamily_family {#1} {#2}
  \tl_set_eq:NN \rmdefault \g_@@_rmfamily_family
  \use:x { \exp_not:n { \DeclareRobustCommand \rmfamily }
   {
    \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
    \exp_not:N \fontfamily { \g_@@_rmfamily_family }
    \exp_not:N \selectfont
   }
  }
  \str_if_eq_x:nnT {\familydefault} {\rmdefault}
    { \tl_set_eq:NN \encodingdefault \l_@@_nfss_enc_tl }
  \@@_setmainfont_hook:nn {#1} {#2}
  \normalfont
  \ignorespaces
 }
```

(*End definition for* `\setmainfont`*. This function is documented on page* ??.)

`\setsansfont`

```
\cs_set:Nn \@@_main_setsansfont:nn
 {
  \fontspec_set_family:Nnn \g_@@_sffamily_family {#1} {#2}
  \tl_set_eq:NN \sfdefault \g_@@_sffamily_family
  \use:x { \exp_not:n { \DeclareRobustCommand \sffamily }
   {
    \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
    \exp_not:N \fontfamily { \g_@@_sffamily_family }
    \exp_not:N \selectfont
   }
  }
  \str_if_eq_x:nnT {\familydefault} {\sfdefault}
    { \tl_set_eq:NN \encodingdefault \l_@@_nfss_enc_tl }
  \@@_setsansfont_hook:nn {#1} {#2}
  \normalfont
  \ignorespaces
 }
```

(*End definition for* `\setsansfont`*. This function is documented on page* ??.)

\setmonofont

```
\cs_set:Nn \@@_main_setmonofont:nn
 {
  \fontspec_set_family:Nnn \g_@@_ttfamily_family {#1} {#2}
  \tl_set_eq:NN \ttdefault \g_@@_ttfamily_family
  \use:x { \exp_not:n { \DeclareRobustCommand \ttfamily }
   {
    \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
    \exp_not:N \fontfamily { \g_@@_ttfamily_family }
    \exp_not:N \selectfont
   }
  }
  \str_if_eq_x:nnT {\familydefault} {\ttdefault}
    { \tl_set_eq:NN \encodingdefault \l_@@_nfss_enc_tl }
  \@@_setmonofont_hook:nn {#1} {#2}
  \normalfont
  \ignorespaces
 }
```

(*End definition for* `\setmonofont`*. This function is documented on page* ??.)

\setmathrm These commands are analogous to `\setmainfont` 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`.

```
\cs_set:Nn \@@_main_setmathrm:nn
  {
⟨XE⟩    \fontspec_set_family:Nnn \g_@@_mathrm_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
    \@@_setmathrm_hook:nn {#1} {#2}
  }
```

(*End definition for* `\setmathrm`*. This function is documented on page* ??.)

\setboldmathrm

```
\cs_set:Nn \@@_main_setboldmathrm:nn
  {
⟨XE⟩    \fontspec_set_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
    \@@_setboldmathrm_hook:nn {#1} {#2}
  }
```

(*End definition for* `\setboldmathrm`*. This function is documented on page* ??.)

\setmathsf

```
\cs_set:Nn \@@_main_setmathsf:nn
  {
⟨XE⟩    \fontspec_set_family:Nnn \g_@@_mathsf_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
    \@@_setmathsf_hook:nn {#1} {#2}
  }
```

(*End definition for* `\setmathsf`*. This function is documented on page* ??.)

\setmathtt

```
\cs_set:Nn \@@_main_setmathtt:nn
  {
⟨XE⟩    \fontspec_set_family:Nnn \g_@@_mathtt_tl {#1} {#2}
⟨LU⟩    \fontspec_set_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
    \@@_setmathtt_hook:nn {#1} {#2}
  }
```

(*End definition for* `\setmathtt`*. This function is documented on page* ??.)

Hooks:

```
\cs_set_eq:NN \@@_setmainfont_hook:nn   \use_none:nn
\cs_set_eq:NN \@@_setsansfont_hook:nn   \use_none:nn
\cs_set_eq:NN \@@_setmonofont_hook:nn   \use_none:nn
\cs_set_eq:NN \@@_setmathrm_hook:nn     \use_none:nn
\cs_set_eq:NN \@@_setmathsf_hook:nn     \use_none:nn
\cs_set_eq:NN \@@_setmathtt_hook:nn     \use_none:nn
\cs_set_eq:NN \@@_setboldmathrm_hook:nn \use_none:nn
```

Hmm, this isn't necessary with unicode-math; oh well:

```
\@onlypreamble\setmathrm
\@onlypreamble\setboldmathrm
\@onlypreamble\setmathsf
\@onlypreamble\setmathtt
```

If the commands above are not executed, then `\rmdefault` (*etc.*) will be used.

```
\tl_set:Nn \g_@@_mathrm_tl {\rmdefault}
\tl_set:Nn \g_@@_mathsf_tl {\sfdefault}
\tl_set:Nn \g_@@_mathtt_tl {\ttdefault}
```

\newfontfamily

This macro takes the arguments of `\fontspec` with a prepended ⟨*instance cmd*⟩. 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.

```
\cs_set:Nn \@@_main_newfontfamily:nnn
 {
  \fontspec_set_family:cnn { g_@@_ \cs_to_str:N #1 _family } {#2} {#3}
  \use:x
   {
    \exp_not:N \DeclareRobustCommand \exp_not:N #1
     {
      \exp_not:N \fontfamily { \use:c {g_@@_ \cs_to_str:N #1 _family} }
      \exp_not:N \fontencoding { \l_@@_nfss_enc_tl }
      \exp_not:N \selectfont
     }
   }
 }
```

(*End definition for* `\newfontfamily`*. This function is documented on page* ??.)

\newfontface

`\newfontface` uses the fact that if the argument to `BoldFont`, etc., is empty (*i.e.*, `BoldFont={}`), then no bold font is searched for.

```
\cs_set:Nn \@@_main_newfontface:nnn
 {
  \newfontfamily #1 [ BoldFont={},ItalicFont={},SmallCapsFont={},#2 ] {#3}
 }
```

(*End definition for* `\newfontface`. *This function is documented on page* ??.)

7.2 Font feature selection

\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.

```
\cs_set:Nn \@@_main_defaultfontfeatures:nnn
  {
    \IfNoValueTF {#2}
     { \@@_set_default_features:nn {#1} {#3} }
     { \@@_set_font_default_features:nnn {#1} {#2} {#3} }
    \ignorespaces
  }
\cs_new:Nn \@@_set_default_features:nn
  {
    \IfBooleanTF {#1} \clist_put_right:Nn \clist_set:Nn
      \g_@@_default_fontopts_clist {#2}
  }
```

The optional argument #2 specifies font identifier(s). Branch for either (a) single token input such as `\rmdefault`, or (b) otherwise assume its a fontname. In that case, strip spaces and file extensions and lower-case to ensure consistency.

```
\cs_new:Nn \@@_set_font_default_features:nnn
 {
  \clist_map_inline:nn {#2}
   {
    \tl_if_single:nTF {##1}
     { \tl_set:No \l_@@_tmp_tl { \cs:w g_@@_ \cs_to_str:N ##1 _family\cs_end: } }
     { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }

    \IfBooleanTF {#1}
     {
      \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
       { \tl_clear:N \l_@@_tmpb_tl }
      \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
      \prop_gput:NVV   \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
     }
     {
      \tl_if_empty:nTF {#3}
       { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
       { \prop_put:NVn    \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
     }
```

```
    }
  }
```

(*End definition for* `\defaultfontfeatures`. *This function is documented on page* ??.)

`\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 \@@_select_font_family:nn), so this means that the only added features to the font are strictly those specified by this command.

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

```
\cs_set:Nn \@@_main_addfontfeatures:n
 {
⟨debug⟩  \typeout{^^J::::::::::::::::::::::::::::::::^^J: addfontfeatures}
  \fontspec_if_fontspec_font:TF
   {
    \group_begin:
      \keys_set_known:nnN {fontspec-addfeatures} {#1} \l_@@_tmp_tl
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {options}  \l_@@_options_tl
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {fontname} \l_@@_fontname_tl
      \bool_set_true:N \l_@@_disable_defaults_bool
⟨debug⟩          \typeout{ \@@_select_font_family:nn { \l_@@_options_tl , #1 } {\l_@@_fontnam
      \use:x
       {
        \@@_select_font_family:nn
          { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
       }
    \group_end:
    \fontfamily\l_fontspec_family_tl\selectfont
   }
   {
    \@@_warning:nx {addfontfeatures-ignored} {#1}
   }
  \ignorespaces
 }
```

(*End definition for* `\addfontfeatures`. *This function is documented on page* ??.)

7.3 Defining new font features

`\newfontfeature` `\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.

```
\cs_set:Nn \@@_main_newfontfeature:nn
 {
  \keys_define:nn { fontspec }
```

```
    {
     #1 .code:n =
      {
       \@@_update_featstr:n {#2}
      }
    }
  }
```

(*End definition for* `\newfontfeature`. *This function is documented on page* ??.)

`\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.

```
\cs_set:Nn \@@_main_newAATfeature:nnnn
 {
  \keys_if_exist:nnF { fontspec } {#1}
    { \@@_define_aat_feature_group:n {#1} }

  \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
    { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }

  \@@_define_aat_feature:nnnn {#1}{#2}{#3}{#4}
 }
```

(*End definition for* `\newAATfeature`. *This function is documented on page* ??.)

`\newopentypefeature` 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.

```
\cs_set:Nn \@@_main_newopentypefeature:nnn
 {
  \keys_if_exist:nnF { fontspec / options } {#1}
    { \@@_define_opentype_feature_group:n {#1} }

  \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
    { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }

  \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
    {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {}
 }
\cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
\cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
  {
    \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
  }
```

(*End definition for* `\newopentypefeature`. *This function is documented on page* ??.)

`\aliasfontfeature` User commands for renaming font features and font feature options.

```
\cs_set:Nn \@@_main_aliasfontfeature:nn
 {
⟨debug⟩ \typeout{::::::::::::::::::::^^J:: aliasfontfeature{#1}{#2}}
  \bool_set_false:N \l_@@_alias_bool
```

```
  \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
   {
     \keys_if_exist:nnT {##1} {#1}
      {
⟨debug⟩ \typeout{:::: Key~exists~##1~/~#1}
        \bool_set_true:N \l_@@_alias_bool
        \keys_define:nn {##1}
          { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
      }
   }

  \bool_if:NF \l_@@_alias_bool
    { \@@_warning:nx {rename-feature-not-exist} {#1} }
 }
```

(*End definition for* `\aliasfontfeature`*. This function is documented on page* ??.)

`\aliasfontfeatureoption`

```
\cs_set:Nn \@@_main_aliasfontfeatureoption:nnn
 {
  \bool_set_false:N \l_@@_alias_bool

  \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
   {
     \keys_if_exist:nnT { ##1 / #1 } {#2}
      {
⟨debug⟩ \typeout{:::: Keyval~exists~##1~/~#1~=~#2}
        \bool_set_true:N \l_@@_alias_bool
        \keys_define:nn { ##1 / #1 }
          { #3 .code:n = { \keys_set:nn {##1} { #1 = {#2} } } }
      }

     \keys_if_exist:nnT { ##1 / #1 } {#2Reset}
      {
⟨debug⟩ \typeout{:::: Keyval~exists~##1~/~#1~=~#2Reset}
        \keys_define:nn { ##1 / #1 }
          { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
      }

     \keys_if_exist:nnT { ##1 / #1 } {#2Off}
      {
⟨debug⟩ \typeout{:::: Keyval~exists~##1~/~#1~=~#2Off}
        \keys_define:nn { ##1 / #1 }
          { #3Off .code:n = { \keys_set:nn {##1} { #1 = {#2Off} } } }
      }
   }

  \bool_if:NF \l_@@_alias_bool
    { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
 }
```

(*End definition for* `\aliasfontfeatureoption`*. This function is documented on page* ??.)

\DeclareFontsExtensions dfont would never be uppercase, right?

```
\cs_set:Nn \@@_main_DeclareFontsExtensions:n
 {
  \clist_set:Nn \l_@@_extensions_clist { #1 }
  \tl_remove_all:Nn \l_@@_extensions_clist {~}
 }
\DeclareFontsExtensions{.otf,.ttf,.OTF,.TTF,.ttc,.TTC,.dfont}
```

(*End definition for* `\DeclareFontsExtensions`. *This function is documented on page* ??.)

\IfFontFeatureActiveTF

```
\cs_set:Nn \@@_main_IfFontFeatureActiveTF:nnn
  {
⟨debug⟩     \typeout{^^J:::::::::::::::::::::::::::::::::::::::::::::::::}
⟨debug⟩     \typeout{:IfFontFeatureActiveTF \exp_not:n{{#1}{#2}{#3}}}
    \@@_if_font_feature:nTF {#1} {#2} {#3}
  }
\prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
  {
    \tl_gclear:N \g_@@_single_feat_tl
    \group_begin:
      \@@_font_suppress_not_found_error:
      \@@_init:
      \bool_set_true:N \l_@@_ot_bool
      \bool_set_true:N \l_@@_never_check_bool
      \bool_set_false:N \l_@@_firsttime_bool
      \clist_clear:N \l_@@_fontfeat_clist
      \@@_get_features:Nn \l_@@_rawfeatures_sclist {#1}
    \group_end:

⟨debug⟩     \typeout{:::> \exp_not:N\l_@@_rawfeatures_sclist->~{\l_@@_rawfeatures_sclist}}
⟨debug⟩     \typeout{:::> \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}

    \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }
      {
        \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
          { \prg_return_true: } { \prg_return_false: }
      }
  }
```

(*End definition for* `\IfFontFeatureActiveTF`. *This function is documented on page* ??.)

8 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.

```
\prg_new_conditional:Nnn \fontspec_if_fontspec_font: {TF,T,F}
 {
  \cs_if_exist:cTF {g_@@_fontinfo_ \f@family _prop} \prg_return_true: \prg_return_false:
 }
```

(*End definition for* `\fontspec_if_fontspec_font:TF`*. This function is documented on page* ??.)

`\fontspec_if_aat_feature:nnTF` Conditional to test if the currently selected font contains the AAT feature (`#1`,`#2`).

```
\prg_new_conditional:Nnn \fontspec_if_aat_feature:nn {TF,T,F}
 {
  \fontspec_if_fontspec_font:TF
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_atsui_bool
     {
      \@@_make_AAT_feature_string:NnnTF \font {#1} {#2}
        \prg_return_true: \prg_return_false:
     }
     {
      \prg_return_false:
     }
   }
   {
    \prg_return_false:
   }
 }
```

(*End definition for* `\fontspec_if_aat_feature:nnTF`*. This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \fontspec_if_opentype: {TF,T,F}
 {
  \fontspec_if_fontspec_font:TF
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool \prg_return_true: \prg_return_false:
   }
   {
    \prg_return_false:
   }
 }
```

(*End definition for* `\fontspec_if_opentype:TF`*. This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \fontspec_if_feature:n {TF,T,F}
```

```
{
 \fontspec_if_fontspec_font:TF
  {
   \@@_set_font_type:N \font
   \bool_if:NTF \l_@@_ot_bool
    {
     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmp_tl
     \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}

     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-num} \l_@@_tmp_tl
     \int_set:Nn \l_@@_language_int {\l_@@_tmp_tl}

     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag}  \l_fontspec_script_tl
     \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag}    \l_fontspec_lang_tl

     \@@_check_ot_feat:NnTF \font {#1} {\prg_return_true:} {\prg_return_false:}
    }
    {
     \prg_return_false:
    }
  }
  {
   \prg_return_false:
  }
}
```

(*End definition for* `\fontspec_if_feature:nTF`*. This function is documented on page* ??.)

`\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:nTF {latn} {ROM} {pnum} {True} {False}` Returns false if the font is not loaded by fontspec or is not an OpenType font.

```
\prg_new_conditional:Nnn \fontspec_if_feature:nnn {TF,T,F}
{
 \fontspec_if_fontspec_font:TF
  {
   \@@_set_font_type:N \font
   \bool_if:NTF \l_@@_ot_bool
    {
     \@@_iv_str_to_num:Nn \l_@@_script_int {#1}
     \@@_iv_str_to_num:Nn \l_@@_language_int {#2}
     \@@_check_ot_feat:NnTF \font {#3} \prg_return_true: \prg_return_false:
    }
    { \prg_return_false: }
  }
  { \prg_return_false: }
}
```

(*End definition for* `\fontspec_if_feature:nnnTF`*. This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \fontspec_if_script:n {TF,T,F}
 {
  \fontspec_if_fontspec_font:TF
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool
     {
      \@@_check_script:NnTF \font {#1} \prg_return_true: \prg_return_false:
     }
     { \prg_return_false: }
   }
   { \prg_return_false: }
 }
```

(*End definition for* `\fontspec_if_script:nTF`*. This function is documented on page* ??.)

`\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
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool
     {
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmpf_tl
      \int_set:Nn \l_@@_script_int {\l_@@_tmpf_tl}
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag}  \l_fontspec_script_tl

      \@@_check_lang:NnTF \font {#1} \prg_return_true: \prg_return_false:
     }
     { \prg_return_false: }
   }
   { \prg_return_false: }
 }
```

(*End definition for* `\fontspec_if_language:nTF`*. This function is documented on page* ??.)

`\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 {cyrl} {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
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool
     {
      \tl_set:Nn \l_fontspec_script_tl {#1}
      \@@_iv_str_to_num:Nn \l_@@_script_int {#1}
      \@@_check_lang:NnTF \font {#2} \prg_return_true: \prg_return_false:
```

```
      }
      { \prg_return_false: }
    }
    { \prg_return_false: }
  }
```

(*End definition for* `\fontspec_if_language:nnTF`. *This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F}
 {
  \fontspec_if_fontspec_font:TF
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool
     {
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag}  \l_@@_tmp_tl
      \str_if_eq:nVTF {#1}  \l_@@_tmp_tl
        {\prg_return_true:} {\prg_return_false:}
     }
     { \prg_return_false: }
   }
   { \prg_return_false: }
 }
```

(*End definition for* `\fontspec_if_current_script:nTF`. *This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F}
 {
  \fontspec_if_fontspec_font:TF
   {
    \@@_set_font_type:N \font
    \bool_if:NTF \l_@@_ot_bool
     {
      \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag}  \l_@@_tmp_tl
      \str_if_eq:nVTF {#1} \l_@@_tmp_tl
        {\prg_return_true:} {\prg_return_false:}
     }
     { \prg_return_false: }
   }
   { \prg_return_false: }
 }
```

(*End definition for* `\fontspec_if_current_language:nTF`. *This function is documented on page* ??.)

`\fontspec_set_family:Nnn`

`#1` : family
`#2` : fontspec features
`#3` : font name

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

We want to store the actual name of the font family within the ⟨*family*⟩ 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 `\@@_select_font_family:nn`, which may change in the future.

```
\cs_new:Nn \fontspec_set_family:Nnn
 {
  \tl_set:Nn \l_@@_family_label_tl { #1 }
  \@@_select_font_family:nn {#2}{#3}
  \tl_set_eq:NN #1 \l_fontspec_family_tl
 }
\cs_generate_variant:Nn \fontspec_set_family:Nnn {c}
```

(*End definition for* `\fontspec_set_family:Nnn`*. This function is documented on page* ??.)

`\fontspec_set_fontface:NNnn`

```
\cs_new:Nn \fontspec_set_fontface:NNnn
 {
  \tl_set:Nn \l_@@_family_label_tl { #1 }
  \@@_select_font_family:nn {#3}{#4}
  \font #1 = \fontname \l_fontspec_font \scan_stop:
  \tl_set_eq:NN #2 \l_fontspec_family_tl
 }
```

(*End definition for* `\fontspec_set_fontface:NNnn`*. This function is documented on page* ??.)

`\fontspec_font_if_exist:n`

```
\prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
  {
    \group_begin:
      \@@_init:
      \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
      \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
        { \group_end: \prg_return_true: }
        { \group_end: \prg_return_false:  }
  }
\cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF
```

(*End definition for* `\fontspec_font_if_exist:n`*. This function is documented on page* ??.)

`\fontspec_if_current_feature:nTF` Test whether the currently loaded font is using the specified raw OpenType feature tag `#1`.

```
\prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
  {
    \exp_args:Nxx \tl_if_in:nnTF
      { \fontname\font } { \tl_to_str:n {#1} }
      { \prg_return_true: } { \prg_return_false: }
  }
```

(*End definition for* `\fontspec_if_current_feature:nTF`*. This function is documented on page* ??.)

\fontspec_if_small_caps:TF

```
\prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
  {
    \@@_if_merge_shape:nTF {sc}
      {
        \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
      }
      {
        \tl_set:Nn \l_@@_smcp_shape_tl {sc}
      }

    \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
      {
        \tl_if_eq:ccTF
          { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
          { \f@encoding/\f@family/\f@series/\updefault }
          { \prg_return_false: }
          { \prg_return_true:  }
      }
      { \prg_return_false: }
  }
```

(*End definition for* `\fontspec_if_small_caps:TF`. *This function is documented on page* ??.)

9 Internals

9.1 The main function for setting fonts

\@@_select_font_family: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_fontspec_family_tl`. The TEX '`\font`' command is (globally) stored in `\l_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.

Some often-used variables to know about:

- `\l_fontspec_fontname_tl` is used as the generic name of the font being defined.
- `\l_@@_fontid_tl` is the unique identifier of the font with all its features.
- `\l_@@_fontname_up_tl` is the font specifically to be used as the upright font.
- `\l_@@_basename_tl` is the (immutable) original argument used for `*`-replacing.
- `\l_fontspec_font` is the plain TEX font of the upright font requested.

```
\cs_new_protected:Nn \@@_select_font_family:nn
 {
⟨debug⟩\typeout{^^J^^J::::::::::::::::::::::::::::::^^J:: fontspec_select:nn~ {#1}~ {#2} }
  \group_begin:
  \@@_font_suppress_not_found_error:
```

```
  \@@_init:

  \@@_sanitise_fontname:Nn \l_fontspec_fontname_tl    {#2}
  \@@_sanitise_fontname:Nn \l_@@_fontname_up_tl {#2}
  \@@_sanitise_fontname:Nn \l_@@_basename_tl          {#2}

  \@@_if_detect_external:nT {#2}
   { \keys_set:nn {fontspec-preparse-external} {Path} }

  \keys_set_known:nn {fontspec-preparse-cfg} {#1}

  \@@_init_ttc:n {#2}
  \@@_load_external_fontoptions:Nn \l_fontspec_fontname_tl {#2}

  \@@_extract_all_features:n {#1}
  \tl_set:Nx \l_@@_fontid_tl { \tl_to_str:N \l_fontspec_fontname_tl-:-\tl_to_str:N \l_@@_all
⟨debug⟩\typeout{fontid: \l_@@_fontid_tl}

  \@@_preparse_features:
  \@@_load_font:
  \@@_set_scriptlang:
  \@@_get_features:Nn \l_@@_rawfeatures_sclist {}
  \bool_set_false:N \l_@@_firsttime_bool

  \@@_save_family_needed:nTF {#2}
   {
     \@@_save_family:nn {#1} {#2}
⟨debug⟩  \@@_warning:nxx {defining-font} {#1} {#2}
   }
   {
⟨debug⟩  \typeout{Font~ family~ already~ defined.}
   }
  \group_end:
 }
```

(*End definition for* `\@@_select_font_family:nn`. *This function is documented on page* ??.)

`\fontspec_select:nn` This old name has been used by 3rd party packages so for compatibility:

```
\cs_set_eq:NN \fontspec_select:nn \@@_select_font_family:nn %% deprecated, for compatibility
```

(*End definition for* `\fontspec_select:nn`. *This function is documented on page* ??.)

`\@@_sanitise_fontname:Nn` Assigns font name `#2` to token list variable `#1` and strips extension(s) from it in the case of an external font. We strip spaces for luatex for consistency with luaotfload, although I'm not sure this is necessary any more. At one stage this also lowercased the name, but this step has been removed unless someone can remind me why it was necessary.

```
\cs_new:Nn \@@_sanitise_fontname:Nn
 {
  \tl_set:Nx #1 {#2}
⟨luatex⟩  \tl_remove_all:Nn #1 {~}
  \clist_map_inline:Nn \l_@@_extensions_clist
```

```
    {
      \tl_if_in:NnT #1 {##1}
        {
          \tl_remove_once:Nn #1 {##1}
          \tl_set:Nn \l_@@_extension_tl {##1}
          \clist_map_break:
        }
    }
  }
```

(*End definition for* `\@@_sanitise_fontname:Nn`*. This function is documented on page* ??.)

`\@@_if_detect_external:nT` Check if either the fontname ends with a known font extension.

```
\prg_new_conditional:Nnn \@@_if_detect_external:n {T}
 {
⟨debug⟩  \typeout{:: @@_if_detect_external:n  { \exp_not:n {#1} } }
  \clist_map_inline:Nn \l_@@_extensions_clist
   {
    \bool_set_false:N \l_@@_tmpa_bool
    \exp_args:Nx % <- this should be handled earlier
    \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
      { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
   }
  \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
 }
```

(*End definition for* `\@@_if_detect_external:nT`*. This function is documented on page* ??.)

`\@@_init_ttc:n` For TTC fonts we assume they will be loading the italic/bold fonts from the same file, so prepopulate the fontnames to avoid needing to do it manually.

```
\cs_new:Nn \@@_init_ttc:n
 {
  \str_if_eq_x:nnT { \str_lower_case:f {\l_@@_extension_tl} } {.ttc}
   {
    \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl   {#1}
    \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl   {#1}
    \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
   }
 }
```

(*End definition for* `\@@_init_ttc:n`*. This function is documented on page* ??.)

`\@@_load_external_fontoptions:Nn` Load a possible `.fontspec` font configuration file. This file could set font-specific options for the font about to be loaded.

```
\cs_new:Nn \@@_load_external_fontoptions:Nn
  {
    \bool_if:NT \l_@@_fontcfg_bool
      {
⟨debug⟩  \typeout{:: @@_load_external_fontoptions:Nn \exp_not:N #1 {#2} }
        \@@_sanitise_fontname:Nn #1 {#2}
        \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
        \tl_remove_all:Nn \l_@@_ext_filename_tl {~}
```

```
        \prop_if_in:NVF \g_@@_fontopts_prop #1
         {
          \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
           { \file_input:n { \l_@@_ext_filename_tl } }
         }
      }
  }
```

(*End definition for* `\@@_load_external_fontoptions:Nn`*. This function is documented on page* ??.)

`\@@_extract_all_features:`

```
\cs_new:Nn \@@_extract_all_features:n
 {
⟨debug⟩  \typeout{:: @@_extract_all_features:n { \unexpanded {#1} } }
  \bool_if:NTF \l_@@_disable_defaults_bool
   {
    \clist_set:Nx \l_@@_all_features_clist {#1}
   }
   {
    \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
     { \clist_clear:N \l_@@_fontopts_clist }

    \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
     { \clist_clear:N \l_@@_family_fontopts_clist }
    \tl_clear:N \l_@@_family_label_tl

    \clist_set:Nx \l_@@_all_features_clist
     {
      \g_@@_default_fontopts_clist,
      \l_@@_family_fontopts_clist,
      \l_@@_fontopts_clist,
      #1
     }
   }
 }
```

(*End definition for* `\@@_extract_all_features:`*. This function is documented on page* ??.)

`\@@_preparse_features:` `#1` : feature options
`#2` : font name

Perform the (multi-step) feature parsing process.

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.

```
\cs_new:Nn \@@_preparse_features:
 {
⟨debug⟩  \typeout{:: @@_preparse_features:}
```

Detect if external fonts are to be used, possibly automatically, and parse fontspec features for bold/italic fonts and their features.

```

  \@@_keys_set_known:nxN {fontspec-preparse-external}
```

```
    { \l_@@_all_features_clist }
    \l_@@_keys_leftover_clist

```

When `\l_fontspec_fontname_tl` is augmented with a prefix or whatever to create the name of the upright font (`\l_@@_fontname_up_tl`), this latter is the new 'general font name' to use.

```
  \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
  \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
    \l_@@_keys_leftover_clist
  \@@_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
    \l_@@_fontfeat_clist
 }
```

(*End definition for* `\@@_preparse_features:`*. This function is documented on page* ??.)

`\@@_load_font:`

```
\cs_new:Nn \@@_load_font:
 {
⟨debug⟩\typeout{:: @@_load_font}
⟨debug⟩\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_
  \@@_primitive_font_set:Nnn \l_fontspec_font
     { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
  \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {\l_@@_font
  \@@_set_font_type:N \l_fontspec_font
⟨debug⟩\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl
  \@@_primitive_font_gset:Nnn \l_fontspec_font
     { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } {} } {\f@size pt}
  \l_fontspec_font % this is necessary for LuaLaTeX to check the scripts properly
 }
```

(*End definition for* `\@@_load_font:`*. This function is documented on page* ??.)

`\@@_construct_font_call:nn` Constructs the complete font invocation. `#1` : Base name
`#2` : Extension
`#3` : TTC Index
`#4` : Renderer
`#5` : Optical size
`#6` : Font features

We check if ⟨*Font features*⟩ are empty and if so don't add in the separator colon.

```
\cs_set:Nn \@@_construct_font_call:nnnnnn
 {
⟨xetexx⟩  " \@@_fontname_wrap:n { #1 #2 #3 }
⟨luatex⟩  " \@@_fontname_wrap:n { #1 #2 } #3
    #4 #5
    \str_if_eq_x:nnF {#6}{} {:#6} "
 }
```

In practice, we don't use the six-argument version, since most arguments are constructed on-the-fly:

```
\cs_set:Nn \@@_construct_font_call:nn
 {
  \@@_construct_font_call:nnnnnn
```

```
    {#1}
    \l_@@_extension_tl
    \l_@@_ttc_index_tl
    \l_fontspec_renderer_tl
    \l_@@_optical_size_tl
    {#2}
  }
```

(*End definition for* `\@@_construct_font_call:nn`*. This function is documented on page* ??.)

`\@@_font_is_file:`
`\@@_font_is_name:`

The `\@@_fontname_wrap:n` command takes the font name and either passes it through unchanged or wraps it in the syntax for loading a font 'by filename'. XeTeX's syntax is followed since luaotfload provides compatibility.

```
\cs_new:Nn \@@_font_is_name:
  {
    \cs_set_eq:NN \@@_fontname_wrap:n \use:n
  }
\cs_new:Nn \@@_font_is_file:
  {
    \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
  }
```

(*End definition for* `\@@_font_is_file:` *and* `\@@_font_is_name:`*. These functions are documented on page* ??.)

`\@@_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.

```
\cs_new:Nn \@@_set_scriptlang:
 {
  \bool_if:NT \l_@@_firsttime_bool
   {
    \tl_if_empty:NTF \l_@@_script_name_tl
     {
      \@@_check_script:NnTF \l_fontspec_font {latn}
       {
        \tl_set:Nn \l_@@_script_name_tl {Latin}
        \tl_if_empty:NT \l_@@_lang_name_tl
         {
          \tl_set:Nn \l_@@_lang_name_tl {Default}
         }
        \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
        \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
       }
       {
        \@@_info:n {no-scripts}
       }
     }
     {
      \tl_if_empty:NT \l_@@_lang_name_tl
       {
        \tl_set:Nn \l_@@_lang_name_tl {Default}
       }
      \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
```

```
      \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
    }
  }
}
```

(*End definition for* `\@@_set_scriptlang:`. *This function is documented on page* ??.)

`\@@_get_features:Nn` 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.

Do not set the colour if not explicitly spec'd else `\color` (using specials) will not work.

```
\cs_set:Nn \@@_get_features:Nn
 {
⟨debug⟩  \typeout{:: @@_get_features:Nn \exp_not:N #1 { \exp_not:n {#2} } }
  \@@_init_fontface:
  \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#2}
    \l_@@_keys_leftover_clist
  \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
⟨*xetexx⟩
  \bool_if:NTF \l_@@_ot_bool
    {
⟨debug⟩  \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clis
    %  \tracingall
      \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
    %  \ERORROR
    }
    {
⟨debug⟩  \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_
      \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
        { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
    }
⟨/xetexx⟩
⟨*luatex⟩
⟨debug⟩  \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clis
  \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
⟨/luatex⟩

  \tl_if_empty:NF \l_@@_mapping_tl
    { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }

  \str_if_eq_x:nnF { \l_@@_hexcol_tl \l_@@_opacity_tl }
                   { \g_@@_hexcol_tl \g_@@_opacity_tl }
    { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }

  \tl_set_eq:NN #1 \l_@@_rawfeatures_sclist
 }
```

(*End definition for* `\@@_get_features:Nn`. *This function is documented on page* ??.)

`\@@_save_family_needed:nTF` 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.

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.

```
\prg_new_conditional:Nnn \@@_save_family_needed:n {TF}
 {

⟨debug⟩  \typeout{save~ family:~ #1}
⟨debug⟩  \typeout{== fontid_tl: "\l_@@_fontid_tl".}

  \tl_if_exist:cF {g_@@_UID_\l_@@_fontid_tl}
    {
      \tl_new:c {g_@@_UID_\l_@@_fontid_tl}
    }

  \tl_if_exist:NT \l_@@_nfss_fam_tl
    {
      \tl_set_eq:cN {g_@@_UID_\l_@@_fontid_tl} \l_@@_nfss_fam_tl
    }

  \tl_if_empty:cT {g_@@_UID_\l_@@_fontid_tl}
   {
    % The font name is fully expanded, in case it's defined in terms of macros, before havin
    \tl_set:Nx \l_@@_tmp_tl {#1}
    \tl_remove_all:Nn \l_@@_tmp_tl {~}

    \cs_if_exist:cTF {g_@@_family_ \l_@@_tmp_tl _int}
     { \int_gincr:c  {g_@@_family_ \l_@@_tmp_tl _int} }
     { \int_new:c    {g_@@_family_ \l_@@_tmp_tl _int} }

    \tl_gset:cx {g_@@_UID_\l_@@_fontid_tl}
     {
      \l_@@_tmp_tl ( \int_use:c {g_@@_family_ \l_@@_tmp_tl _int} )
     }
   }
  \tl_gset:Nv \l_fontspec_family_tl {g_@@_UID_\l_@@_fontid_tl}
  \cs_if_exist:cTF {g_@@_fontinfo_ \l_fontspec_family_tl _prop}
    \prg_return_false: \prg_return_true:
 }
```

(*End definition for* `\@@_save_family_needed:nTF`*. This function is documented on page* ??.)

`\@@_save_family:nn` Saves the relevant font information for future processing.

```
\cs_new:Nn \@@_save_family:nn
  {
    \@@_save_fontinfo:n {#2}
    \@@_find_autofonts:
    \DeclareFontFamily{\l_@@_nfss_enc_tl}{\l_fontspec_family_tl}{}
    \@@_set_faces:
    \@@_info:nxx {defining-font} {#1} {#2}
  }
```

(*End definition for* `\@@_save_family:nn`*. This function is documented on page* ??.)

`\@@_save_fontinfo:n` Saves the relevant font information for future processing.

```
\cs_new:Nn \@@_save_fontinfo:n
 {
  \prop_new:c    {g_@@_fontinfo_ \l_fontspec_family_tl _prop}
  \prop_gput:cnx {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {fontname} { #1 }
  \prop_gput:cnx {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {options}  { \l_@@_all_feature
  \prop_gput:cnx {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {fontdef}
   {
    \@@_construct_font_call:nn {\l_fontspec_fontname_tl}
      { \l_@@_pre_feat_sclist \l_@@_rawfeatures_sclist }
   }
  \prop_gput:cnV {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {script-num} \l_@@_script_int
  \prop_gput:cnV {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {lang-num} \l_@@_language_int
  \prop_gput:cnV {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {script-tag} \l_fontspec_scrip
  \prop_gput:cnV {g_@@_fontinfo_ \l_fontspec_family_tl _prop} {lang-tag} \l_fontspec_lang_tl
 }
```

(*End definition for* `\@@_save_fontinfo:n`*. This function is documented on page* ??.)

9.2 Setting font shapes in a family

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 `\bfdefault` is redefined to `b`, all bold shapes defined by this package will also be assigned to `b`.

The combination shapes are searched first because they use information that may be redefined in the single cases. E.g., if no bold font is specified then `set_autofont` will attempt to set it. This has subtle/small ramifications on the logic of choosing the bold italic font.

`\@@_find_autofonts:`

```
\cs_new:Nn \@@_find_autofonts:
 {
  \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
   {
    \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
    \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
    \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspec_fontname_tl} {/BI}
   }

  \bool_if:NF \l_@@_nobf_bool
   {
    \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspec_fontname_tl} {/B}
   }

  \bool_if:NF \l_@@_noit_bool
   {
    \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspec_fontname_tl} {/I}
   }

  \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
 }
```

(*End definition for* `\@@_find_autofonts:`. *This function is documented on page* ??.)

`\@@_set_faces:`

```
\cs_new:Nn \@@_set_faces:
 {
  \@@_add_nfssfont:nnnn \mddefault \updefault \l_fontspec_fontname_tl      \l_@@_fontfeat_up
  \@@_add_nfssfont:nnnn \bfdefault \updefault \l_@@_fontname_bf_tl   \l_@@_fontfeat_bf_clist
  \@@_add_nfssfont:nnnn \mddefault \itdefault \l_@@_fontname_it_tl   \l_@@_fontfeat_it_clist
  \@@_add_nfssfont:nnnn \mddefault \sldefault \l_@@_fontname_sl_tl   \l_@@_fontfeat_sl_clist
  \@@_add_nfssfont:nnnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfit_cli
  \@@_add_nfssfont:nnnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfsl_cli

  \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
 }
\cs_new:Nn \@@_set_faces_aux:nnnnn
 {
  \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
  \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
 }
```

(*End definition for* `\@@_set_faces:`. *This function is documented on page* ??.)

`\fontspec_complete_fontname:Nn` 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`").

```
\cs_set:Nn \fontspec_complete_fontname:Nn
 {
  \tl_set:Nx #1 {#2}
  \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
⟨luatex⟩  \tl_remove_all:Nn #1 {~}
 }
```

(*End definition for* `\fontspec_complete_fontname:Nn`. *This function is documented on page* ??.)

`\@@_add_nfssfont:nnnn`

`#1` : series
`#2` : shape
`#3` : fontname
`#4` : fontspec features

```
\cs_new:Nn \@@_add_nfssfont:nnnn
 {
  \tl_set:Nx \l_@@_this_font_tl {#3}

  \tl_if_empty:xTF {#4}
   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }

  \tl_if_empty:NF \l_@@_this_font_tl
   {
    \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
     { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
   }
 }
```

(*End definition for* `\@@_add_nfssfont:nnnn`*. This function is documented on page* ??.)

9.2.1 Fonts

`\@@_set_font_type:N` Now check if the font is to be rendered with ATSUI or Harfbuzz. 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.

```
\cs_new:Nn \@@_set_font_type:N
 {
⟨debug⟩  \typeout{:: @@_set_font_type:}
⟨*xetexx⟩
  \bool_set_false:N \l_@@_tfm_bool
  \bool_set_false:N \l_@@_atsui_bool
  \bool_set_false:N \l_@@_ot_bool
  \bool_set_false:N \l_@@_mm_bool
  \bool_set_false:N \l_@@_graphite_bool
  \ifcase\XeTeXfonttype #1
    \bool_set_true:N \l_@@_tfm_bool
  \or
    \bool_set_true:N \l_@@_atsui_bool
    \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/AAT} }
    \ifnum\XeTeXcountvariations #1 > \c_zero
      \bool_set_true:N \l_@@_mm_bool
    \fi
  \or
    \bool_set_true:N \l_@@_ot_bool
    \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/OT} }
  \or
    \bool_set_true:N \l_@@_graphite_bool
    \tl_if_empty:NT \l_fontspec_renderer_tl { \tl_set:Nn \l_fontspec_renderer_tl {/GR} }
  \fi
⟨/xetexx⟩
```

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.

LuaTeX only supports one:

```
⟨*luatex⟩
  \bool_set_true:N \l_@@_ot_bool
⟨/luatex⟩
 }
```

(*End definition for* `\@@_set_font_type:N`*. This function is documented on page* ??.)

`\@@_set_autofont:Nnn` `#1` : Font name tl
`#2` : Base font name
`#3` : Font name modifier

This function looks for font with ⟨*name*⟩ and ⟨*modifier*⟩ `#2#3`, and if found (i.e., different to font with name `#2`) stores it in tl `#1`. A modifier is something like `/B` to look for a bold font, for example.

We can't match external fonts in this way (in XeTeX anyway; todo: test with LuaTeX). If ⟨*font name tl*⟩ is not empty, then it's already been specified by the user so abort. If ⟨*Base font name*⟩ is not given, we also abort for obvious reasons.

If ⟨*font name tl*⟩ is empty, then proceed. If not found, ⟨*font name tl*⟩ remains empty. Otherwise, we have a match.

```
\cs_new:Nn \@@_set_autofont:Nnn
 {
  \bool_if:NF \l_@@_external_bool
   {
  \tl_if_empty:xF {#2}
   {
    \tl_if_empty:NT #1
     {
      \@@_if_autofont:nnTF {#2} {#3}
       { \tl_set:Nx #1 {#2#3} }
       { \@@_info:nx {no-font-shape} {#2#3} }
     }
   }
   }
 }

\prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
 {
  \@@_primitive_font_set:Nnn \l_tmpa_font { \@@_construct_font_call:nn {#1} {}  } {\f@size p
  \@@_primitive_font_set:Nnn \l_tmpb_font { \@@_construct_font_call:nn {#1#2} {} } {\f@size
  \str_if_eq_x:nnTF { \fontname \l_tmpa_font } { \fontname \l_tmpb_font }
   { \prg_return_false: }
   { \prg_return_true: }
 }
```

(*End definition for* `\@@_set_autofont:Nnn`*. This function is documented on page* ??.)

`\@@_make_font_shapes:Nnnnn`

`#1` : Font name
`#2` : Font series
`#3` : Font shape
`#4` : Font features
`#5` : Size features

This macro eventually uses `\DeclareFontShape` to define the font shape in question.

```
\cs_new:Nn \@@_make_font_shapes:Nnnnn
 {
  \group_begin:
    \@@_keys_set_known:nxN {fontspec-preparse-external} { #4 } \l_@@_leftover_clist
    \@@_load_fontname:n {#1}
    \@@_declare_shape:nnxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
  \group_end:
 }

```

```
\cs_new:Nn \@@_load_fontname:n
 {
⟨debug⟩    \typeout{:: @@_load_fontname:n {#1} }
    \@@_load_external_fontoptions:Nn \l_fontspec_fontname_tl {#1}
    \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
     { \clist_clear:N \l_@@_fontopts_clist }
    \@@_primitive_font_set:Nnn \l_fontspec_font { \@@_construct_font_call:nn {\l_fontspec_fo
    \@@_primitive_font_if_null:NT \l_fontspec_font { \@@_error:nx {font-not-found} {#1} }
 }
```

(*End definition for* `\@@_make_font_shapes:Nnnnn`*. This function is documented on page* ??.)

`\@@_declare_shape:nnnn`

`#1` : Font series
`#2` : Font shape
`#3` : Font features
`#4` : Size features

Wrapper for `\DeclareFontShape`. And finally the actual font shape declaration using `\l_@@_nfss_tl` defined above. `\l_@@_postadjust_tl` is defined in various places to deal with things like the hyphenation character and interword spacing.

The main part is to loop through `SizeFeatures` arguments, which are of the form

`SizeFeatures={{<one>},{<two>},{<three>}}`.

```
\cs_new:Nn \@@_declare_shape:nnnn
 {
⟨debug⟩\typeout{=~ declare_shape:~{\l_fontspec_fontname_tl}~{#1}~{#2}}
  \tl_clear:N \l_@@_nfss_tl
  \tl_clear:N \l_@@_nfss_sc_tl
  \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspec_fontname_tl

  \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }

  \@@_declare_shapes_normal:nn {#1} {#2}
  \@@_declare_shapes_smcaps:nn {#1} {#2}
  \@@_declare_shape_slanted:nn {#1} {#2}
  \@@_declare_shape_loginfo:nn {#1} {#2}
 }
\cs_generate_variant:Nn \@@_declare_shape:nnnn {nnxx}
```

(*End definition for* `\@@_declare_shape:nnnn`*. This function is documented on page* ??.)

`\@@_setup_single_size:nn`

```
\cs_new:Nn \@@_setup_single_size:nn
  {
    \tl_clear:N \l_@@_size_tl
    \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed

    \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
      \l_@@_sizing_leftover_clist
    \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
⟨debug⟩\typeout{==~ size:~\l_@@_size_tl}

    % "normal"
```

```
    \@@_load_fontname:n {\l_@@_sizedfont_tl}
    \@@_setup_nfss:Nnnn \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
⟨debug⟩    \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}

    % small caps
    \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist

    \bool_if:NF \l_@@_nosc_bool
     {
      \tl_if_empty:NTF \l_@@_fontname_sc_tl
       {
        \@@_make_smallcaps:TF
         {
⟨debug⟩\typeout{====~Small~ caps~ found.}
          \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
         }
         {
⟨debug⟩\typeout{====~Small~ caps~ not~ found.}
          \bool_set_true:N \l_@@_nosc_bool
         }
       }
       { \@@_load_fontname:n {\l_@@_fontname_sc_tl} }% local for each size
     }

    \bool_if:NF \l_@@_nosc_bool
     {
      \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
        {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
     }
  }
```

(*End definition for* `\@@_setup_single_size:nn`*. This function is documented on page* ??.)

`\@@_setup_nfss:Nnnn`

```
\cs_new:Nn \@@_setup_nfss:Nnnn
 {
⟨debug⟩\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}

  \@@_get_features:Nn \l_@@_rawfeatures_sclist { #2 , #3 , #4 }
⟨debug⟩\typeout{====~Gathered~features:~\l_@@_rawfeatures_sclist}

  \tl_put_right:Nx #1
   {
    <\l_@@_size_tl> \l_@@_scale_tl
      \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
        { \l_@@_pre_feat_sclist \l_@@_rawfeatures_sclist }
   }
 }
```

(*End definition for* `\@@_setup_nfss:Nnnn`*. This function is documented on page* ??.)

`\@@_declare_shapes_normal:nn`

```
\cs_new:Nn \@@_declare_shapes_normal:nn
  {
    \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl} {\l_fontspec_family_tl}
      {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
  }
```

(*End definition for* `\@@_declare_shapes_normal:nn`*. This function is documented on page* ??.)

`\@@_declare_shapes_smcaps:nn`

```
\cs_new:Nn \@@_declare_shapes_smcaps:nn
  {
    \tl_if_empty:NF \l_@@_nfss_sc_tl
     {
      \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl} {\l_fontspec_family_tl} {#1}
        { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
     }
  }

\cs_new:Nn \@@_combo_sc_shape:n
  {
    \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
         { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
         { \scdefault }
  }
```

(*End definition for* `\@@_declare_shapes_smcaps:nn`*. This function is documented on page* ??.)

`\@@_DeclareFontShape:nnnnnn`

```
\cs_new:Nn \@@_DeclareFontShape:nnnnnn
 {
⟨debug⟩\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
  \group_begin:
    \normalsize
    \cs_undefine:c {#1/#2/#3/#4/\f@size}
  \group_end:
  \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
 }
\cs_generate_variant:Nn \@@_DeclareFontShape:nnnnnn {xxxxxx}
```

`\@@_declare_shape_slanted:nn`

This extra stuff for the slanted shape substitution is a little bit awkward. We define the slanted shape to be a synonym for it when (a) we're defining an italic font, but also (b) when the default slanted shape isn't 'it'. (Presumably this turned up once in a test and I realised it caused problems. I doubt this would happen much.)

We should test when a slanted font has been specified and not run this code if so, but the `\@@_set_slanted:` code will overwrite this anyway if necessary.

```
\cs_new:Nn \@@_declare_shape_slanted:nn
 {
  \bool_if:nT
   {
     \str_if_eq_x_p:nn {#2} {\itdefault}  &&
    !(\str_if_eq_x_p:nn {\itdefault} {\sldefault})
   }
```

```
    {
     \@@_DeclareFontShape:xxxxxx {\l_@@_nfss_enc_tl}{\l_fontspec_family_tl}{#1}{\sldefault}
       {<->ssub*\l_fontspec_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
    }
  }
```

Lastly some informative messaging.

\@@_declare_shape_loginfo:nn

```
\cs_new:Nn \@@_declare_shape_loginfo:nn
 {
  \tl_gput_right:Nx \l_fontspec_defined_shapes_tl
   {
    \exp_not:n { \\ }
    -~ \exp_not:N \str_case:nn {#1/#2}
     {
       {\mddefault/\updefault} {'normal'~}
       {\bfdefault/\updefault} {'bold'~}
       {\mddefault/\itdefault} {'italic'~}
       {\mddefault/\sldefault} {'slanted'~}
       {\bfdefault/\itdefault} {'bold~ italic'~}
       {\bfdefault/\sldefault} {'bold~ slanted'~}
     } (#1/#2)~
    with~ NFSS~ spec.:~
    \l_@@_nfss_tl
    \exp_not:n { \\ }
    -~ \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
     {
       {\mddefault/\scdefault} {'small~ caps'~}
       {\bfdefault/\scdefault} {'bold~ small~ caps'~}
       {\mddefault/\itscdefault} {'italic~ small~ caps'~}
       {\bfdefault/\itscdefault} {'bold~ italic~ small~ caps'~}
       {\mddefault/\slscdefault} {'slanted~ small~ caps'~}
       {\bfdefault/\slscdefault} {'bold~ slanted~ small~ caps'~}
     }~( #1 / \@@_combo_sc_shape:n {#2} )~
    with~ NFSS~ spec.:~
    \l_@@_nfss_sc_tl
    \tl_if_empty:fF {\l_@@_postadjust_tl}
     {
      \exp_not:N \\ and~ font~ adjustment~ code: \exp_not:N \\ \l_@@_postadjust_tl
     }
   }
 }
```

Maybe `\str_if_eq_x:nnF` would be better?

9.2.2 Features

These are the features always applied to a font selection before other features.

\l_@@_pre_feat_sclist

```
\tl_set:Nn \l_@@_pre_feat_sclist
⟨*xetexx⟩
 {
  \bool_if:NT \l_@@_ot_bool
   {
```

```
    \tl_if_empty:NF \l_fontspec_script_tl
     {
      script   = \l_fontspec_script_tl ;
      language = \l_fontspec_lang_tl   ;
     }
   }
 }
⟨/xetexx⟩
⟨*luatex⟩
 {
  mode     = \l_fontspec_mode_tl   ;
  \tl_if_empty:NF \l_fontspec_script_tl
   {
    script   = \l_fontspec_script_tl ;
    language = \l_fontspec_lang_tl   ;
   }
 }
⟨/luatex⟩
```

This macro checks if the font contains small caps.

\@@_make_ot_smallcaps:TF

```
⟨luatex⟩\cs_set:Nn \@@_make_smallcaps:TF
⟨xetexx⟩\cs_set:Nn \@@_make_ot_smallcaps:TF
 {
  \@@_check_ot_feat:NnTF \l_fontspec_font {smcp} {#1} {#2}
 }
⟨*xetexx⟩
\cs_set:Nn \@@_make_smallcaps:TF
 {
  \bool_if:NTF \l_@@_ot_bool
   { \@@_make_ot_smallcaps:TF {#1} {#2} }
   {
     \bool_if:NT \l_@@_atsui_bool
      { \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {3}{3} {#1} {#2} }
   }
 }
⟨/xetexx⟩
```

\@@_update_featstr:n

`\l_@@_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_new:Nn \@@_update_featstr:n
  {
⟨debug⟩            \typeout{:::: @@_update_featstr:n {#1}}
    \bool_if:NF \l_@@_firsttime_bool
      {
        \tl_gset:Nx \g_@@_single_feat_tl { #1 }
⟨debug⟩            \typeout{::::~ Adding~ feature.}
        \tl_gput_right:Nx  \l_@@_rawfeatures_sclist {#1;}
      }
  }
```

\@@_remove_clashing_featstr:n

```
\cs_new:Nn \@@_remove_clashing_featstr:n
  {
⟨debug⟩    \typeout{:::: @@_remove_clashing_featstr:n {#1}}
    \clist_map_inline:nn {#1}
      {
⟨debug⟩        \typeout{::::~ Removing~ feature~ "##1;"}
        \tl_gremove_all:Nn \l_@@_rawfeatures_sclist {##1;}
      }
  }
```

9.3 Initialisation

Initialisations that need to occur once per fontspec font invocation. (Some of these may be redundant. Check whether they're assigned to globally or not.)

\@@_init:

```
\cs_set:Npn \@@_init:
 {
⟨debug⟩  \typeout{:: @@_init:}
  \bool_set_false:N \l_@@_ot_bool
  \bool_set_true:N \l_@@_firsttime_bool
  \@@_font_is_name:
  \tl_clear:N \l_@@_font_path_tl
  \tl_clear:N \l_@@_optical_size_tl
  \tl_clear:N \l_@@_ttc_index_tl
  \tl_clear:N \l_fontspec_renderer_tl
  \tl_clear:N \l_fontspec_defined_shapes_tl
  \tl_clear:N \g_@@_curr_series_tl
  \tl_gset_eq:NN \l_@@_nfss_enc_tl \g_fontspec_encoding_tl

⟨*luatex⟩
  \tl_set:Nn \l_fontspec_mode_tl {node}
  \int_set:Nn \luatex_prehyphenchar:D { `\- } % fixme
  \int_zero:N \luatex_posthyphenchar:D        % fixme
  \int_zero:N \luatex_preexhyphenchar:D       % fixme
  \int_zero:N \luatex_postexhyphenchar:D      % fixme
⟨/luatex⟩
 }
```

Executed in `\@@_get_features:Nn`.

\@@_init_fontface:

```
\cs_new:Nn \@@_init_fontface:
  {
    \tl_clear:N \l_@@_rawfeatures_sclist
    \tl_clear:N \l_@@_scale_tl
    \tl_set_eq:NN \l_@@_opacity_tl \g_@@_opacity_tl
    \tl_set_eq:NN \l_@@_hexcol_tl \g_@@_hexcol_tl
    \tl_set_eq:NN \l_@@_postadjust_tl \g_@@_postadjust_tl
    \tl_clear:N \l_@@_wordspace_adjust_tl
    \tl_clear:N \l_@@_punctspace_adjust_tl
  }
```

9.4 Miscellaneous

\@@_iv_str_to_num:Nn

This macro takes a four character string and converts it to the numerical representation required for XeTeX OpenType script/language/feature purposes. The output is stored in #1.

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 s and anything beyond four chars is snipped. The \@empty s then are used to reconstruct the spaces in the string to number calculation.

For backwards compatibility this code also strips a leading + or -.

```
\cs_set:Nn \@@_iv_str_to_num:Nn
  {
    \@@_strip_leading_sign:Nw #1#2 \q_nil
  }
\cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
  {
    \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
      { \@@_iv_str_to_num:w #1 \q_nil #3   \c_empty_tl \c_empty_tl \q_nil }
      { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
  }
\cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
 {
  \int_set:Nn #1
   {
      `#2 * "1000000
    + `#3 * "10000
    + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
    + \ifx \c_empty_tl #5 32 \else `#5 \fi
   }
 }
\cs_generate_variant:Nn \@@_iv_str_to_num:Nn {No}
```

10 OpenType definitions code

\@@_define_opentype_feature_group:n

```
\cs_new:Nn \@@_define_opentype_feature_group:n
  {
    \keys_define:nn {fontspec-opentype} { #1 .multichoice: }
  }
```

\@@_define_opentype_feature:nnnnn

#1 : Feature key
#2 : Feature option val
#3 : Check feature — leave empty for no check
#4 : Exact tag string to activate — leave empty for disable only
#5 : Tags to remove (clist)

```
\cs_new:Nn \@@_feat_prop_add:nn
  {
    \tl_if_empty:nF {#1}
     {
```

```
      \prop_if_in:NnF \g_@@_OT_features_prop {#1}
        {
          \prop_gput:Nnn \g_@@_OT_features_prop {#1} {#2}
        }
    }
  }
\cs_new:Nn \@@_define_opentype_feature:nnnnn
  {
    \@@_feat_prop_add:nn {#3} {#1\,=\,#2}
      \tl_if_empty:nTF {#4}
        {
          \keys_define:nn {fontspec-opentype}
            {
              #1/#2 .code:n =
                { \@@_remove_clashing_featstr:n {#5} }
            }
        }
        {
          \keys_define:nn {fontspec-opentype}
            {
              #1/#2 .code:n =
                {
⟨debug⟩           \typeout{::::::::fontspec-opentype~#1/#2~=~#3/#4/#5}
                  \@@_make_OT_feature:nnn {#3} {#4} {#5}
                }
            }
        }
  }
```

\@@_define_opentype_onoffreset:nnnnn

#1 : Feature key
#2 : Feature option val
#3 : Check feature
#4 : Tag prefix to activate: +#4 = on, -#4 = off.
#5 : Tags to remove in the on case (clist)

```
\cs_new:Nn \@@_feat_off:n {#1Off}
\cs_new:Nn \@@_feat_reset:n {#1Reset}

\cs_new:Nn \@@_define_opentype_onoffreset:nnnnn
 {
  \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {+#4} {#5}
  \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_off:n   {#2} } {#3} {-#4} 
  \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4,
 }
```

\@@_define_opentype_onreset:nnnnn

#1 : Feature key
#2 : Feature option val
#3 : Check feature
#4 : Exact tag string to activate
#5 : Tags to remove (clist)

```
\cs_new:Nn \@@_define_opentype_onreset:nnnnn
 {
```

```
  \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} {#2} {#3} {#4} {#5}
  \exp_args:Nnx \@@_define_opentype_feature:nnnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
}
```

10.1 Adding features when loading fonts

When remove clashing features,

1. remove the feature being added (to avoid duplicates);
2. remove the inverse of the feature (to avoid cancellation);
3. finally remove all clashing features.

```
\cs_new:Nn \@@_make_OT_feature:nnn
  {
⟨debug⟩  \typeout{:: @@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }

    \bool_set_true:N \l_@@_proceed_bool
    \bool_set_true:N  \l_@@_check_feat_bool

    \tl_if_empty:nT {#1} { \bool_set_false:N \l_@@_check_feat_bool }
    \bool_if:NT \l_@@_check_feat_bool
      {
        \@@_check_ot_feat:NnF \l_fontspec_font {#1}
          {
            \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
            \bool_set_false:N \l_@@_proceed_bool
          }
      }

    \bool_if:NT \l_@@_proceed_bool
      {
        \exp_args:Nx \@@_remove_clashing_featstr:n
          { #2 , \@@_swap_plus_minus:n {#2} , #3 }

        \@@_update_featstr:n {#2}
      }
  }
\cs_generate_variant:Nn \@@_make_OT_feature:nnn {xxx}
\cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:Nq #1 \q_nil }
\cs_new:Npn \@@_swap_plus_minus_aux:Nq #1#2 \q_nil
  { \str_case:nn {#1} { {+} {-#2} {-} {+#2} } }
```

(*End definition for* `\@@_DeclareFontShape:nnnnnn` *and others. These functions are documented on page* ??.)

`\@@_check_script:NnTF` This macro takes an OpenType script tag and checks if it exists in the current font. The output boolean is `\@tempswatrue`. `\l_@@_script_int` is used to store the number corresponding to the script tag string.

```
\prg_new_conditional:Nnn \@@_check_script:Nn {TF}
  {
    \bool_if:NTF \l_@@_never_check_bool
```

```
      { \prg_return_true: }
⟨*xetexx⟩
 {
  \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
  \int_set:Nn \l_tmpb_int { \XeTeXOTcountscripts #1 }
  \int_zero:N \l_tmpa_int
  \bool_set_false:N \l__fontspec_check_bool
  \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
   {
    \ifnum \XeTeXOTscripttag #1 \l_tmpa_int = \l_@@_strnum_int
      \bool_set_true:N \l__fontspec_check_bool
      \int_set:Nn \l_tmpa_int {\l_tmpb_int}
    \else
      \int_incr:N \l_tmpa_int
    \fi
   }
  \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
 }
⟨/xetexx⟩
⟨*luatex⟩
  {
    \cs_if_eq:NNTF #1 \font
      { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
      { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
    \directlua{fontspec.check_ot_script("\l_@@_tmp_tl", "#2")}
    \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
  }
⟨/luatex⟩
 }
```

(*End definition for* `\@@_check_script:NnTF`. *This function is documented on page* ??.)

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

```
\prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
  {
    \bool_if:NTF \l_@@_never_check_bool
      { \prg_return_true: }
⟨*xetexx⟩
 {
  \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
  \int_set:Nn \l_tmpb_int
   { \XeTeXOTcountlanguages #1 \l_@@_script_int }
  \int_zero:N \l_tmpa_int
  \bool_set_false:N \l__fontspec_check_bool
  \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
   {
    \ifnum\XeTeXOTlanguagetag #1 \l_@@_script_int \l_tmpa_int = \l_@@_strnum_int
      \bool_set_true:N \l__fontspec_check_bool
      \int_set:Nn \l_tmpa_int {\l_tmpb_int}
```

```
    \else
      \int_incr:N \l_tmpa_int
    \fi
   }
  \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
 }
⟨/xetexx⟩
⟨*luatex⟩
  {
    \cs_if_eq:NNTF #1 \font
      { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
      { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
    \directlua
      {
        fontspec.check_ot_lang( "\l_@@_tmp_tl", "#2", "\l_fontspec_script_tl" )
      }
    \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
  }
⟨/luatex⟩
  }
```

(*End definition for* `\@@_check_lang:NnTF`*. This function is documented on page* ??.)

`\@@_check_ot_feat:NnTF` This macro takes an OpenType feature tag and checks if it exists in the current font/script/language. `\l_@@_strnum_int` is used to store the number corresponding to the feature tag string. The script used is whatever's held in `\l_@@_script_int`. By default, that's the number corresponding to '`latn`'. The language used is `\l_@@_language_int`, by default `0`, the 'default language'.

```
\prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
  {
    \bool_if:NTF \l_@@_never_check_bool
      { \prg_return_true: }
⟨*xetexx⟩
 {
⟨debug⟩\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
  \int_set:Nn \l_tmpb_int
   {
    \XeTeXOTcountfeatures #1
                          \l_@@_script_int
                          \l_@@_language_int
   }
  \@@_iv_str_to_num:Nn \l_@@_strnum_int {#2}
  \int_zero:N \l_tmpa_int
  \bool_set_false:N \l_@@_check_bool
  \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
   {
    \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int
         \l_tmpa_int =\l_@@_strnum_int
      \bool_set_true:N \l_@@_check_bool
      \int_set:Nn \l_tmpa_int {\l_tmpb_int}
    \else
      \int_incr:N \l_tmpa_int
    \fi
```

```
    }
   \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
 }
⟨/xetexx⟩
⟨*luatex⟩
  {
⟨debug⟩\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
    \cs_if_eq:NNTF #1 \font
      { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
      { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
    \directlua
      {
        fontspec.check_ot_feat(
                          "\l_@@_tmp_tl", "#2",
                          "\l_fontspec_lang_tl", "\l_fontspec_script_tl"
                         )
      }
    \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
  }
⟨/luatex⟩
  }
```

(*End definition for* `\@@_check_ot_feat:NnTF`. *This function is documented on page* ??.)

10.2 OpenType feature information

```
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access~All~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above-base~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above-base~Mark~Positioning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above-base~Substitutions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative~Fractions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below-base~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below-base~Mark~Positioning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below-base~Substitutions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {case}{Case-Sensitive~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph~Composition~/~Decompositio
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct~Form~After~Ro}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cjct}{Conjunct~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual~Ligatures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered~CJK~Punctuation}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpsp}{Capital~Spacing}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cswh}{Contextual~Swash}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive~Positioning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character~Variant~$N$}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite~Capitals~From~Capitals}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small~Capitals~From~Capitals}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary~Ligatures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless~Forms}
```

```
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final~Glyph~on~Line~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal~Forms~\#2}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal~Forms~\#3}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened~accent~forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate~Half~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal~Kana~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical~Ligatures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo~Kanji~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfbd}{Left~Bounds}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard~Ligatures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading~Jamo~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining~Figures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {locl}{Localized~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right~alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right~mirrored~forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark~Positioning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial~Forms~\#2}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical~Greek}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark~to~Mark~Positioning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark~Positioning~via~Substitutio
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate~Annotation~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC~Kanji~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle~Figures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical~Bounds}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional~Alternate~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite~Capitals}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional~Kana}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional~Figures}
```

```
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre-Base~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre-base~Substitutions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post-base~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post-base~Substitutions}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required~Contextual~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required~Ligatures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right~Bounds}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left~alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left~mirrored~forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby~Notation~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required~Variation~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific~Inferiors}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical~size}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small~Capitals}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smpl}{Simplified~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic~Set~$N$}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math~script~style~alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching~Glyph~Decomposition}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sups}{Superscript}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing~Jamo~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional~Name~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular~Figures}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third~Widths}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate~Vertical~Metrics}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu~Variants}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical~Writing}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate~Vertical~Half~Metrics}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel~Jamo~Forms}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical~Kana~Alternates}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkrn}{Vertical~Kerning}
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional~Alternate~Vertical~
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical~Alternates~and~Rotation
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical~Alternates~for~Rotation
\prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {zero}{Slashed~Zero}
```

11 Graphite/AAT code

\@@_define_aat_feature_group:n

```
\cs_new:Nn \@@_define_aat_feature_group:n
  { \keys_define:nn {fontspec-aat} { #1 .multichoice: } }
```

(*End definition for* `\@@_define_aat_feature_group:n`. *This function is documented on page* ??.)

`\@@_define_aat_feature:nnnn`

```
\cs_new:Nn \@@_define_aat_feature:nnnn
 {
  \keys_define:nn {fontspec-aat}
   {
    #1/#2 .code:n = { \@@_make_AAT_feature:nn {#3}{#4} }
   }
 }
```

(*End definition for* `\@@_define_aat_feature:nnnn`. *This function is documented on page* ??.)

`\@@_make_AAT_feature:nn`

```
\cs_new:Nn \@@_make_AAT_feature:nn
 {
  \tl_if_empty:nTF {#1}
   { \@@_warning:n {aat-feature-not-exist} }
   {
     \@@_make_AAT_feature_string:NnnTF \l_fontspec_font {#1}{#2}
      {
       \@@_update_featstr:n {\l_fontspec_feature_string_tl}
     }
     { \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2} }
   }
 }
```

(*End definition for* `\@@_make_AAT_feature:nn`. *This function is documented on page* ??.)

`\@@_make_AAT_feature_string:NnnTF` 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 47).

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 XeTeX 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 `!` to denote the switch.

Finally, save out the complete feature string in `\l_fontspec_feature_string_tl`.

```
\prg_new_conditional:Nnn \@@_make_AAT_feature_string:Nnn {TF,T,F}
 {
  \tl_set:Nx \l_tmpa_tl { \XeTeXfeaturename #1 #2 }
  \tl_if_empty:NTF \l_tmpa_tl
   { \prg_return_false: }
   {
    \int_compare:nTF { \XeTeXisexclusivefeature #1 #2 > 0 }
     {
      \tl_set:Nx \l_tmpb_tl {\XeTeXselectorname #1 #2\space #3}
     }
     {
      \int_if_even:nTF {#3}
```

```
        {
         \tl_set:Nx \l_tmpb_tl {\XeTeXselectorname #1 #2\space #3}
        }
        {
         \tl_set:Nx \l_tmpb_tl
          {
           \XeTeXselectorname #1 #2\space \numexpr#3-1\relax
          }
         \tl_if_empty:NF \l_tmpb_tl { \tl_put_left:Nn \l_tmpb_tl {!} }
        }
      }
      \tl_if_empty:NTF \l_tmpb_tl
       { \prg_return_false: }
       {
        \tl_set:Nx \l_fontspec_feature_string_tl { \l_tmpa_tl = \l_tmpb_tl }
        \prg_return_true:
       }
    }
  }
```

(*End definition for* `\@@_make_AAT_feature_string:NnnTF`. *This function is documented on page* ??.)

12 Font loading (keyval) definitions

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

```
\clist_set:Nn \g_@@_all_keyval_modules_clist
  {
    fontspec, fontspec-opentype, fontspec-aat,
    fontspec-preparse, fontspec-preparse-cfg, fontspec-preparse-external, fontspec-preparse-
    fontspec-renderer
  }
\cs_new:Nn \@@_keys_define_code:nnn
  {
   \keys_define:nn {#1} { #2 .code:n = {#3} }
  }
```

For catching features that cannot be used in `\addfontfeatures`:

```
\cs_new:Nn \@@_aff_error:n
  {
    \@@_keys_define_code:nnn {fontspec-addfeatures} {#1}
      { \@@_error:nx {not-in-addfontfeatures} {#1} }
  }
```

12.0.1 Pre-parsing naming information

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

Don't load font config file

```
\@@_keys_define_code:nnn {fontspec-preparse-cfg} {IgnoreFontspecFile}
  {
    \bool_set_false:N \l_@@_fontcfg_bool
  }
\@@_keys_define_code:nnn {fontspec-preparse-external} {IgnoreFontspecFile}
  {
    \bool_set_false:N \l_@@_fontcfg_bool
  }
```

`Path` 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.

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
 {
  \bool_set_true:N \l_@@_nobf_bool
  \bool_set_true:N \l_@@_noit_bool
  \bool_set_true:N \l_@@_external_bool
  \tl_set:Nn \l_@@_font_path_tl {#1}
  \@@_font_is_file:
⟨*xetexx⟩
  \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
⟨/xetexx⟩
 }
\aliasfontfeature{Path}{ExternalLocation}
\@@_keys_define_code:nnn {fontspec} {Path} {}
```

(*End definition for* `Path`*. This function is documented on page* ??.)

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

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {Extension}
 {
  \tl_set:Nn \l_@@_extension_tl {#1}
  \bool_if:NF \l_@@_external_bool
   {
    \keys_set:nn {fontspec-preparse-external} {Path}
   }
 }
\tl_clear:N \l_@@_extension_tl
\@@_keys_define_code:nnn {fontspec} {Extension} {}
```

12.0.2 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-renderer}
```

```
 {
  Renderer .choices:nn =
   {AAT,ICU,OpenType,Graphite,Full,Basic}
   {
    \int_compare:nTF {\l_keys_choice_int <= 4} {
⟨*xetexx⟩
      \tl_set:Nx \l_fontspec_renderer_tl
        {
          \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
        }
      \tl_gset:Nx \g_@@_single_feat_tl { \l_fontspec_renderer_tl }
⟨/xetexx⟩
⟨*luatex⟩
      \@@_warning:nx {only-xetex-feature} {Renderer=AAT/OpenType/Graphite}
⟨/luatex⟩
     }
     {
⟨*xetexx⟩
      \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic}
⟨/xetexx⟩
⟨*luatex⟩
      \tl_set:Nx \l_fontspec_mode_tl
        {
          \int_case:nn \l_keys_choice_int { 5 {node} 6 {base} }
        }
      \tl_gset:Nx \g_@@_single_feat_tl { mode=\l_fontspec_mode_tl }
⟨/luatex⟩
     }
   }
 }
```

OpenType script/language See later for the resolutions from fontspec features to OpenType definitions.

```
\@@_keys_define_code:nnn {fontspec-preparse} {Script}
 {
⟨xetexx⟩    \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
  \tl_set:Nn \l_@@_script_name_tl {#1}
 }
```

Exactly the same:

```
\@@_keys_define_code:nnn {fontspec-preparse} {Language}
 {
⟨xetexx⟩    \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
  \tl_set:Nn \l_@@_lang_name_tl {#1}
 }
```

TTC font index

```
\@@_keys_define_code:nnn {fontspec-preparse} {FontIndex}
 {
  \str_if_eq_x:nnF { \str_lower_case:f {\l_@@_extension_tl} } {.ttc}
    { \@@_warning:n {font-index-needs-ttc} }
```

```
⟨xetexx⟩  \tl_set:Nn \l_@@_ttc_index_tl {:#1}
⟨luatex⟩  \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
 }
\@@_keys_define_code:nnn {fontspec} {FontIndex}
 {
⟨xetexx⟩  \tl_set:Nn \l_@@_ttc_index_tl {:#1}
⟨luatex⟩  \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
 }
```

12.0.3 Bold/italic choosing options

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

Bold (NFSS) Series By default, fontspec uses the default bold series, `\bfdefault`. We want to be able to make this extensible.

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSeries}
 {
  \tl_gset:Nx \g_@@_curr_series_tl { #1 }
  \seq_gput_right:Nx \g_@@_bf_series_seq { #1 }
 }
```

Fonts Upright:

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {UprightFont}
 {
  \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse-external} {FontName}
 {
  \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
 }
```

Bold:

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldFont}
 {
  \tl_if_empty:nTF {#1}
   {
    \bool_set_true:N \l_@@_nobf_bool
   }
   {
    \bool_set_false:N \l_@@_nobf_bool
    \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}

    \seq_if_empty:NT \g_@@_bf_series_seq
     {
      \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
      \seq_put_right:Nx \g_@@_bf_series_seq {\bfdefault}
     }
    \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
     { \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl }

```

```
⟨debug⟩\typeout{Setting~bold~font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}

    \prop_put:NxV \l_@@_nfss_prop
     {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl

   }
 }
```

Same for italic:

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {ItalicFont}
 {
  \tl_if_empty:nTF {#1}
   {
    \bool_set_true:N \l_@@_noit_bool
   }
   {
    \bool_set_false:N \l_@@_noit_bool
    \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
   }
 }
```

Simpler for bold+italic & slanted:

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldItalicFont}
 {
  \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse-external} {SlantedFont}
 {
  \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSlantedFont}
 {
  \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
 }
```

Small caps isn't pre-parsed because it can vary with others above:

```
\@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
 {
  \tl_if_empty:nTF {#1}
   {
    \bool_set_true:N \l_@@_nosc_bool
   }
   {
    \bool_set_false:N \l_@@_nosc_bool
    \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
   }
 }
```

Features

```
\@@_keys_define_code:nnn {fontspec-preparse} {UprightFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_up_clist {#1}
```

```
 }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_bf_clist {#1}

%  \prop_put:NxV \l_@@_nfss_prop
%     {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
 }
\@@_keys_define_code:nnn {fontspec-preparse} {ItalicFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_it_clist {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldItalicFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_bfit_clist {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse} {SlantedFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_sl_clist {#1}
 }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
 {
  \clist_set:Nn \l_@@_fontfeat_bfsl_clist {#1}
 }
```

Note that small caps features can vary by shape, so these in fact *aren't* pre-parsed.

```
\@@_keys_define_code:nnn {fontspec} {SmallCapsFeatures}
 {
  \bool_if:NF \l_@@_firsttime_bool
   {
    \clist_set:Nn \l_@@_fontfeat_sc_clist {#1}
   }
 }
```

paragraphFeatures varying by size

```
\@@_keys_define_code:nnn {fontspec-preparse} {SizeFeatures}
 {
  \clist_set:Nn \l_@@_sizefeat_clist {#1}
  \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
 }
\@@_keys_define_code:nnn {fontspec-preparse-nested} {SizeFeatures}
 {
  \clist_set:Nn \l_@@_sizefeat_clist {#1}
  \tl_if_empty:NT \l_@@_this_font_tl
   { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
 }
\@@_keys_define_code:nnn {fontspec-preparse-nested} {Font}
 {
  \tl_set:Nn \l_@@_this_font_tl {#1}
 }
\@@_keys_define_code:nnn {fontspec} {SizeFeatures}
 {
```

```
  % dummy
 }
\@@_keys_define_code:nnn {fontspec} {Font}
 {
  % dummy
 }
\@@_keys_define_code:nnn {fontspec-sizing} {Size}
 {
  \tl_set:Nn \l_@@_size_tl {#1}
 }
\@@_keys_define_code:nnn {fontspec-sizing} {Font}
 {
  \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
 }
```

12.0.4 Font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```
\@@_keys_define_code:nnn {fontspec-preparse} {NFSSEncoding}
 {
  \tl_gset:Nx \l_@@_nfss_enc_tl { #1 }
 }
```

NFSS family Interactions with other packages will sometimes require setting the NFSS family explicitly. (By default fontspec auto-generates one based on the font name.)

```
\@@_keys_define_code:nnn {fontspec-preparse} {NFSSFamily}
 {
  \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
  \cs_undefine:c {g_@@_UID_\l_@@_fontid_tl}
  \tl_if_exist:NT \l_fontspec_family_tl
   { \cs_undefine:c {g_@@_fontinfo_ \l_fontspec_family_tl _prop} }
 }
```

NFSS series/shape This option looks similar in name but has a very different function.

```
\@@_keys_define_code:nnn {fontspec} {FontFace}
 {
  \tl_set:No \l_@@_arg_tl { \use_iii:nnn #1 }
  \tl_set_eq:NN \l_@@_this_feat_tl \l_@@_arg_tl
  \tl_clear:N \l_@@_this_font_tl
  \int_compare:nT { \clist_count:N \l_@@_arg_tl = 1 }
   {
⟨*debug⟩
    \typeout{FontFace~ parsing:~ one~ clist~ item}
⟨/debug⟩
    \tl_if_in:NnF \l_@@_arg_tl {=}
     {
⟨*debug⟩
```

```
      \typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
⟨/debug⟩
      \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_tl
      \tl_clear:N \l_@@_this_feat_tl
     }
   }

  \@@_add_nfssfont:nnnn
   {\use_i:nnn #1}{\use_ii:nnn #1}{\l_@@_this_font_tl}{\l_@@_this_feat_tl}
 }
```

Scale If the input isn't one of the pre-defined string options, then it's gotta be numerical. `\fontspec_calc_scale:n` does all the work in the auto-scaling cases.

```
\@@_keys_define_code:nnn {fontspec} {Scale}
 {
  \str_case:nnF {#1}
   {
    {MatchLowercase} { \@@_calc_scale:n {5} }
    {MatchUppercase} { \@@_calc_scale:n {8} }
   }
   { \tl_set:Nx \l_@@_scale_tl {#1} }
  \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
 }
```

`\@@_calc_scale:n` 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 XƎTEX).

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.

To begin, change to `\rmfamily` but use internal commands in case csrmfamily has been overwritten. (Note that changing `\rmfamily` with fontspec resets `\encodingdefault` appropriately.)

```
\cs_new:Nn \@@_calc_scale:n
 {
  \group_begin:

    \fontencoding { \encodingdefault }
    \fontfamily { \rmdefault }
    \selectfont

    \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
    \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_fontspec_font

    \tl_gset:Nx \l_@@_scale_tl
      {
        \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
                     \dim_to_fp:n {\l_@@_tmpb_dim} }
      }
```

```
    \@@_info:n {set-scale}
  \group_end:
 }
```

(*End definition for* `\@@_calc_scale:n`*. This function is documented on page* ??.)

`\@@_set_font_dimen:NnN` 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'.

```
\cs_new:Nn \@@_set_font_dimen:NnN
 {
  \dim_set:Nn #1 { \fontdimen #2 #3 }
  \dim_compare:nNnT #1 = {0pt}
   {
    \settoheight #1
     {
      \str_if_eq:nnTF {#3} {\font} \rmfamily #3
      \int_case:nnF #2
       {
         {5} {x} % x-height
         {8} {X} % cap-height
       } {?} % "else" clause; never reached.
     }
   }
 }
```

(*End definition for* `\@@_set_font_dimen:NnN`*. This function is documented on page* ??.)

Inter-word space These options set the relevant `\fontdimen`s for the font being loaded.

```
\@@_keys_define_code:nnn {fontspec} {WordSpace}
 {
  \bool_if:NF \l_@@_firsttime_bool
   { \_fontspec_parse_wordspace:w #1,,,\q_stop }
 }
\@@_aff_error:n {WordSpace}
```

`\_fontspec_parse_wordspace:w` 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_set:Nn \l_@@_wordspace_adjust_tl
     {
      \fontdimen 2 \font = #1 \fontdimen 2 \font
      \fontdimen 3 \font = #1 \fontdimen 3 \font
      \fontdimen 4 \font = #1 \fontdimen 4 \font
     }
   }
```

```
    {
     \tl_set:Nn \l_@@_wordspace_adjust_tl
      {
       \fontdimen 2 \font = #1 \fontdimen 2 \font
       \fontdimen 3 \font = #2 \fontdimen 3 \font
       \fontdimen 4 \font = #3 \fontdimen 4 \font
      }
    }
  }
```

(*End definition for* `\_fontspec_parse_wordspace:w`. *This function is documented on page* ??.)

Punctuation space Scaling factor for the nominal `\fontdimen#7`.

```
\@@_keys_define_code:nnn {fontspec} {PunctuationSpace}
 {
  \str_case_x:nnF {#1}
   {
    {WordSpace}
    {
     \tl_set:Nn \l_@@_punctspace_adjust_tl
      { \fontdimen 7 \font = 0 \fontdimen 2 \font }
    }
    {TwiceWordSpace}
    {
     \tl_set:Nn \l_@@_punctspace_adjust_tl
      { \fontdimen 7 \font = 1 \fontdimen 2 \font }
    }
   }
   {
     \tl_set:Nn \l_@@_punctspace_adjust_tl
     { \fontdimen 7 \font = #1 \fontdimen 7 \font }
   }
 }
\@@_aff_error:n {PunctuationSpace}
```

Secret hook into the font-adjustment code

```
\@@_keys_define_code:nnn {fontspec} {FontAdjustment}
 {
  \tl_put_right:Nx \l_@@_postadjust_tl {#1}
 }
```

Letterspacing

```
\@@_keys_define_code:nnn {fontspec} {LetterSpace}
 {
  \@@_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.

LuaTeX decouples hyphenation from font settings, so only HyphenChar=None works for that engine.

```
\@@_keys_define_code:nnn {fontspec} {HyphenChar}
 {
  \str_if_eq:nnTF {#1} {None}
   {
    \tl_put_right:Nn \l_@@_postadjust_tl
      { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
   }
   {
    \@@_warning:nx {only-xetex-feature} {HyphenChar}

    \tl_if_single:nTF {#1}
     { \tl_set:Nn \l_fontspec_hyphenchar_tl {`#1} }
     { \tl_set:Nn \l_fontspec_hyphenchar_tl { #1} }

    \@@_primitive_font_glyph_if_exist:NnTF \l_fontspec_font {\l_fontspec_hyphenchar_tl}
     {
      \tl_put_right:Nn \l_@@_postadjust_tl
        { \@@_primitive_font_set_hyphenchar:Nn \font { \l_fontspec_hyphenchar_tl } }
     }
     { \@@_error:nx {no-glyph}{#1} }

   }
 }
\@@_aff_error:n {HyphenChar}
```

Color Hooks into pkgxcolor, which names its colours \color@<name>.

```
\@@_keys_define_code:nnn {fontspec} {Color}
 {
  \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
   {
    \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
   }
   {
    \int_compare:nTF { \tl_count:n {#1} == 6 }
     { \tl_set:Nn \l_@@_hexcol_tl {#1} }
     {
      \int_compare:nTF { \tl_count:n {#1} == 8 }
       { \fontspec_parse_colour:viii #1 }
       {
        \bool_if:NF \l_@@_firsttime_bool
         { \@@_warning:nx {bad-colour} {#1} }
       }
     }
   }
 }
\cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
```

```
 {
  \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
  \tl_if_eq:NNF \l_@@_opacity_tl \g_@@_opacity_tl
   {
    \bool_if:NF \l_@@_firsttime_bool
     { \@@_warning:nx {opa-twice-col} {#7#8} }
   }
  \tl_set:Nn \l_@@_opacity_tl {#7#8}
 }
\aliasfontfeature{Color}{Colour}
\@@_keys_define_code:nnn {fontspec} {Opacity}
 {
  \int_set:Nn \l_@@_tmp_int {255}
  \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
  \tl_if_eq:NNF \l_@@_opacity_tl \g_@@_opacity_tl
   {
    \bool_if:NF \l_@@_firsttime_bool
     { \@@_warning:nx {opa-twice} {#1} }
   }
  \tl_set:Nx \l_@@_opacity_tl
   {
     \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
     \int_to_hex:n { \l_@@_tmp_int }
   }
 }
```

Mapping

```
⟨*xetexx⟩
\@@_keys_define_code:nnn {fontspec-aat} {Mapping}
  {
    \tl_set:Nn \l_@@_mapping_tl { #1 }
  }
\@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
  {
    \tl_set:Nn \l_@@_mapping_tl { #1 }
  }
⟨/xetexx⟩
⟨*luatex⟩
\@@_keys_define_code:nnn {fontspec-opentype} {Mapping}
 {
  \str_if_eq:nnTF {#1} {tex-text}
   {
    \@@_warning:n {no-mapping-ligtex}
    \msg_redirect_name:nnn {fontspec} {no-mapping-ligtex} {none}
    \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
   }
   { \@@_warning:n {no-mapping} }
 }
⟨/luatex⟩
```

12.0.5 Continuous font axes

```
\@@_keys_define_code:nnn {fontspec} {Weight}
 {
  \@@_update_featstr:n{weight=#1}
 }
\@@_keys_define_code:nnn {fontspec} {Width}
 {
  \@@_update_featstr:n{width=#1}
 }
\@@_keys_define_code:nnn {fontspec} {OpticalSize}
⟨*xetexx⟩
 {
  \bool_if:NTF \l_@@_ot_bool
   {
    \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
   }
   {
    \bool_if:NT \l_@@_mm_bool
     {
      \@@_update_featstr:n { optical size = #1 }
     }
   }
  \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
   {
    \bool_if:NT \l_@@_firsttime_bool
     { \@@_warning:n {no-opticals} }
   }
 }
⟨/xetexx⟩
⟨*luatex⟩
 {
  \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
 }
⟨/luatex⟩
```

12.0.6 Font transformations

These are to be specified to apply directly to a font shape:

```
\keys_define:nn {fontspec}
 {
  FakeSlant .code:n =
   {
    \@@_update_featstr:n{slant=#1}
   },
  FakeSlant .default:n = {0.2}
}
\keys_define:nn {fontspec}
 {
  FakeStretch .code:n =
   {
    \@@_update_featstr:n{extend=#1}
   },
```

```
  FakeStretch .default:n = {1.2}
}
⟨*xetexx⟩
\keys_define:nn {fontspec}
 {
  FakeBold .code:n =
   {
    \@@_update_featstr:n {embolden=#1}
   },
  FakeBold .default:n = {1.5}
 }
⟨/xetexx⟩
⟨*luatex⟩
\keys_define:nn {fontspec}
 {
  FakeBold .code:n = { \@@_warning:n {fakebold-only-xetex} }
 }
⟨/luatex⟩
```

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 *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.)

```
\keys_define:nn {fontspec}
 {
  AutoFakeSlant .code:n =
   {
    \bool_if:NT \l_@@_firsttime_bool
     {
      \tl_set:Nn \l_@@_fake_slant_tl {#1}
      \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
      \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
      \bool_set_false:N \l_@@_noit_bool

      \tl_if_empty:NF \l_@@_fake_embolden_tl
       {
        \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
         {FakeBold=\l_@@_fake_embolden_tl}
        \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
        \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
       }
     }
   },
  AutoFakeSlant .default:n = {0.2}
}
```

Same but reversed:

```
\keys_define:nn {fontspec}
 {
```

```
  AutoFakeBold .code:n =
   {
    \bool_if:NT \l_@@_firsttime_bool
     {
      \tl_set:Nn \l_@@_fake_embolden_tl {#1}
      \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
      \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
      \bool_set_false:N \l_@@_nobf_bool

      \tl_if_empty:NF \l_@@_fake_slant_tl
       {
        \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
         {FakeSlant=\l_@@_fake_slant_tl}
        \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
        \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
       }
     }
   },
  AutoFakeBold .default:n = {1.5}
}
```

12.0.7 Raw feature string

This allows savvy XeTeX-ers to input font features manually if they have already memorised the OpenType abbreviations and don't mind not having error checking.

```
\@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
 {
  \@@_update_featstr:n {#1}
 }
\@@_keys_define_code:nnn {fontspec-aat} {RawFeature}
 {
  \@@_update_featstr:n {#1}
 }
```

12.1 OpenType feature definitions

```
\@@_feat_prop_add:nn {salt} { Alternate\,=\,$N$ }
\@@_feat_prop_add:nn {nalt} { Annotation\,=\,$N$ }
\@@_feat_prop_add:nn {ornm} { Ornament\,=\,$N$ }
\@@_feat_prop_add:nn {cvNN} { CharacterVariant\,=\,$N$:$M$ }
\@@_feat_prop_add:nn {ssNN} { StylisticSet\,=\,$N$ }
```

12.2 Regular key=val / tag definitions

12.2.1 Ligatures

```
\@@_define_opentype_feature_group:n {Ligatures}
\@@_define_opentype_feature:nnnnn {Ligatures} {ResetAll} {} {}
  {
    +dlig,-dlig,+rlig,-rlig,+liga,-liga,+dlig,-dlig,+clig,-clig,+hlig,-hlig,
⟨xetexx⟩  mapping = tex-text
⟨luatex⟩  +tlig,-tlig
```

```
  }
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Required}      {rlig} {rlig} {}
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Common}        {liga} {liga} {}
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Rare}          {dlig} {dlig} {}
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Discretionary} {dlig} {dlig} {}
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Contextual}    {clig} {clig} {}
\@@_define_opentype_onoffreset:nnnnn {Ligatures} {Historic}      {hlig} {hlig} {}
```

Emulate CM extra ligatures.

```
⟨*xetexx⟩
\keys_define:nn {fontspec-opentype}
  {
    Ligatures / TeX .code:n = { \tl_set:Nn \l_@@_mapping_tl {tex-text} },
    Ligatures / TeXReset .code:n = { \tl_clear:N \l_@@_mapping_tl },
  }
⟨/xetexx⟩
⟨luatex⟩\@@_define_opentype_onreset:nnnnn {Ligatures} {TeX} {} { +tlig } {}
```

12.2.2 Letters

```
\@@_define_opentype_feature_group:n {Letters}
\@@_define_opentype_feature:nnnnn   {Letters} {ResetAll} {} {}
  {
    +case,+smcp,+pcap,+c2sc,+c2pc,+unic,+rand,
    -case,-smcp,-pcap,-c2sc,-c2pc,-unic,-rand
  }
\@@_define_opentype_onoffreset:nnnnn {Letters} {Uppercase} {case} {case} {+smcp,+pcap,+c2sc,
\@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {smcp} {+pcap,+unic,+rand}
\@@_define_opentype_onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic,+rand
\@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+un
\@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+u
\@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {unic} {+rand}
\@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {+unic}
```

12.2.3 Numbers

```
\@@_define_opentype_feature_group:n {Numbers}
\@@_define_opentype_feature:nnnnn   {Numbers} {ResetAll} {} {}
  {
    +tnum,-tnum,
    +pnum,-pnum,
    +onum,-onum,
    +lnum,-lnum,
    +zero,-zero,
    +anum,-anum,
  }
\@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced}   {tnum} {tnum} {+pnum,-pnum}
\@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {pnum} {+tnum,-tnum}
\@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase}    {onum} {onum} {+lnum,-lnum}
\@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase}    {lnum} {lnum} {+onum,-onum}
\@@_define_opentype_onoffreset:nnnnn {Numbers} {SlashedZero}  {zero} {zero} {}
\aliasfontfeatureoption {Numbers} {Monospaced} {Tabular}
```

```
\aliasfontfeatureoption {Numbers} {Lowercase}  {OldStyle}
\aliasfontfeatureoption {Numbers} {Uppercase}  {Lining}
```

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⟩  \@@_define_opentype_onoffreset:nnnnn {Numbers} {Arabic} {anum} {anum} {}
```

12.2.4 Vertical position

```
\@@_define_opentype_feature_group:n  {VerticalPosition}
\@@_define_opentype_feature:nnnnn    {VerticalPosition} {ResetAll} {} {}
  {
    +sups,-sups,
    +subs,-subs,
    +ordn,-ordn,
    +numr,-numr,
    +dnom,-dnom,
    +sinf,-sinf,
  }
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior}           {sups} {sups} {
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior}           {subs} {subs} {
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal}            {ordn} {ordn} {
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator}          {numr} {numr} {
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator}        {dnom} {dnom} {
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {
```

12.2.5 Contextuals

```
\@@_define_opentype_feature_group:n  {Contextuals}
\@@_define_opentype_feature:nnnnn    {Contextuals} {ResetAll} {} {}
  {
    +cswh,-cswh,
    +calt,-calt,
    +init,-init,
    +fina,-fina,
    +falt,-falt,
    +medi,-medi,
  }
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {Swash}       {cswh} {cswh} {}
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {Alternate}   {calt} {calt} {}
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordInitial} {init} {init} {}
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordFinal}   {fina} {fina} {}
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {LineFinal}   {falt} {falt} {}
\@@_define_opentype_onoffreset:nnnnn {Contextuals} {Inner}       {medi} {medi} {}
```

12.2.6 Diacritics

```
\@@_define_opentype_feature_group:n  {Diacritics}
\@@_define_opentype_feature:nnnnn    {Diacritics} {ResetAll} {} {}
  {
    +mark,-mark,
    +mkmk,-mkmk,
    +abvm,-abvm,
```

```
    +blwm,-blwm,
  }
\@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToBase} {mark} {mark} {}
\@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToMark} {mkmk} {mkmk} {}
\@@_define_opentype_onoffreset:nnnnn {Diacritics} {AboveBase}  {abvm} {abvm} {}
\@@_define_opentype_onoffreset:nnnnn {Diacritics} {BelowBase}  {blwm} {blwm} {}
```

12.2.7 Kerning

```
\@@_define_opentype_feature_group:n  {Kerning}
\@@_define_opentype_feature:nnnnn    {Kerning} {ResetAll} {} {}
  {
    +cpsp,-cpsp,
    +kern,-kern,
  }
\@@_define_opentype_onoffreset:nnnnn {Kerning} {Uppercase} {cpsp} {cpsp} {}
\@@_define_opentype_feature:nnnnn    {Kerning} {On}        {kern} {+kern} {-kern}
\@@_define_opentype_feature:nnnnn    {Kerning} {Off}       {kern} {-kern} {+kern}
\@@_define_opentype_feature:nnnnn    {Kerning} {Reset}     {} {} {+kern,-kern}
```

12.2.8 Fractions

```
\@@_define_opentype_feature_group:n  {Fractions}
\@@_define_opentype_feature:nnnnn    {Fractions} {ResetAll} {} {}
  {
    +frac,-frac,
    +afrc,-afrc,
  }
\@@_define_opentype_feature:nnnnn    {Fractions} {On}    {frac} {+frac} {}
\@@_define_opentype_feature:nnnnn    {Fractions} {Off}   {frac} {-frac} {}
\@@_define_opentype_feature:nnnnn    {Fractions} {Reset} {} {} {+frac,-frac}
\@@_define_opentype_onoffreset:nnnnn {Fractions} {Alternate} {afrc} {afrc} {-frac}
```

12.2.9 Style

```
\@@_define_opentype_feature_group:n  {Style}
\@@_define_opentype_feature:nnnnn    {Style} {ResetAll} {} {}
  {
    +salt,-salt,
    +ital,-ital,
    +ruby,-ruby,
    +swsh,-swsh,
    +hist,-hist,
    +titl,-titl,
    +hkna,-hkna,
    +vkna,-vkna,
    +ssty=0,-ssty=0,
    +ssty=1,-ssty=1,
  }
\@@_define_opentype_onoffreset:nnnnn {Style} {Alternate}        {salt} {salt} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {Italic}           {ital} {ital} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {Ruby}             {ruby} {ruby} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {Swash}            {swsh} {swsh} {}
```

```
\@@_define_opentype_onoffreset:nnnnn {Style} {Cursive}          {swsh} {curs} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {Historic}         {hist} {hist} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps}      {titl} {titl} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {HorizontalKana}   {hkna} {hkna} {+vkna,+pkna}
\@@_define_opentype_onoffreset:nnnnn {Style} {VerticalKana}     {vkna} {vkna} {+hkna,+pkna}
\@@_define_opentype_onoffreset:nnnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
\@@_define_opentype_feature:nnnnn    {Style} {MathScript}       {ssty} {+ssty=0} {+ssty=1}
\@@_define_opentype_feature:nnnnn    {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}
```

12.2.10 CJK shape

```
\@@_define_opentype_feature_group:n  {CJKShape}
\@@_define_opentype_feature:nnnnn    {CJKShape} {ResetAll} {} {}
  {
    +trad,-trad,
    +smpl,-smpl,
    +jp78,-jp78,
    +jp83,-jp83,
    +jp90,-jp90,
    +jp04,-jp04,
    +expt,-expt,
    +nlck,-nlck,
  }
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified}  {smpl} {smpl} {+trad,+jp78,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978}     {jp78} {jp78} {+trad,+smpl,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983}     {jp83} {jp83} {+trad,+smpl,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990}     {jp90} {jp90} {+trad,+smpl,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004}     {jp04} {jp04} {+trad,+smpl,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert}      {expt} {expt} {+trad,+smpl,+jp
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC}         {nlck} {nlck} {+trad,+smpl,+jp
```

12.2.11 Character width

```
\@@_define_opentype_feature_group:n  {CharacterWidth}
\@@_define_opentype_feature:nnnnn    {CharacterWidth} {ResetAll} {} {}
  {
    +pwid,-pwid,
    +fwid,-fwid,
    +hwid,-hwid,
    +twid,-twid,
    +qwid,-qwid,
    +palt,-palt,
    +halt,-halt,
  }
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Proportional}          {pwid} {pwid}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Full}                  {fwid} {fwid}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Half}                  {hwid} {hwid}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Third}                 {twid} {twid}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Quarter}               {qwid} {qwid}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateProportional} {palt} {palt}
\@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateHalf}         {halt} {halt}
```

12.2.12 Vertical

According to spec vkrn must also activate vpal if available but for simplicity we don't do that here (yet?).

```
\@@_define_opentype_feature_group:n {Vertical}
\@@_define_opentype_onoffreset:nnnnn {Vertical} {RotatedGlyphs}         {vrt2} {vrt2} {+vrtr
\@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2
\@@_define_opentype_onoffreset:nnnnn {Vertical} {Alternates}            {vert} {vert} {+vrt2
\@@_define_opentype_onoffreset:nnnnn {Vertical} {KanaAlternates}        {vkna} {vkna} {+hkna
\@@_define_opentype_onoffreset:nnnnn {Vertical} {Kerning}               {vkrn} {vkrn} {}
\@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternateMetrics}      {valt} {valt} {+vhal
\@@_define_opentype_onoffreset:nnnnn {Vertical} {HalfMetrics}           {vhal} {vhal} {+valt
\@@_define_opentype_onoffreset:nnnnn {Vertical} {ProportionalMetrics}   {vpal} {vpal} {+valt
```

12.3 OpenType features that need numbering

12.3.1 Alternate

```
\@@_define_opentype_feature_group:n  {Alternate}
\keys_define:nn {fontspec-opentype}
 {
  Alternate .default:n = {0} ,
⟨luatex⟩  Alternate / Random  .code:n =
⟨luatex⟩   { \@@_make_OT_feature:nnn {salt}{ +salt = random }{} } ,
  Alternate / unknown .code:n =
   {
    \clist_map_inline:nn {#1}
      { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }
   }
 }
\aliasfontfeature{Alternate}{StylisticAlternates}
```

12.3.2 Variant / StylisticSet

```
\@@_define_opentype_feature_group:n  {Variant}
\keys_define:nn {fontspec-opentype}
 {
  Variant .default:n = {0} ,
  Variant / unknown .code:n =
   {
    \clist_map_inline:nn {#1}
      {
        \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
      }
   }
 }
\aliasfontfeature{Variant}{StylisticSet}
```

12.3.3 CharacterVariant

```
\@@_define_opentype_feature_group: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
   {
     \@@_make_OT_feature:xxx
       { cv \exp_not:N \two@digits {##1} } { +cv \exp_not:N \two@digits {##1} = ##2 } {}
   }
  \keys_define:nn {fontspec-opentype}
   {
    CharacterVariant / unknown .code:n =
     {
      \clist_map_inline:nn {##1}
       {
        \exp_not:N \fontspec_parse_cv:w
          ####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
       }
     }
   }
 }
```

Possibilities: `a:0:\q_nil` or `a:b:0:\q_nil`.

12.3.4 Annotation

```
\@@_define_opentype_feature_group:n {Annotation}
\keys_define:nn {fontspec-opentype}
 {
  Annotation .default:n = {0} ,
  Annotation / unknown .code:n =
   {
    \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
   }
 }
```

12.3.5 Ornament

```
\@@_define_opentype_feature_group:n  {Ornament}
\keys_define:nn {fontspec-opentype}
 {
  Ornament .default:n = {0} ,
  Ornament / unknown .code:n =
   {
    \@@_make_OT_feature:nnn {ornm} { +ornm=#1 } {}
   }
 }
```

12.4 Script and Language

12.4.1 Script

```
\keys_define:nn { fontspec-opentype } { Script .choice: }
\cs_new:Nn \fontspec_new_script:nn
 {
  \keys_define:nn { fontspec-opentype } { Script / #1 .code:n =
    \bool_set_false:N \l_@@_script_exist_bool
    \clist_map_inline:nn {#2}
```

```
      {
       \@@_check_script:NnTF \l_fontspec_font {####1}
        {
         \tl_set:Nn \l_fontspec_script_tl {####1}
         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
         \bool_set_true:N \l_@@_script_exist_bool
         \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
         \clist_map_break:
        }
        { }
      }
     \bool_if:NF \l_@@_script_exist_bool
      {
       \str_if_eq:nnTF {#1} {Latin}
        {
         \@@_warning:nx {script-not-exist} {#1}
        }
        {
         \@@_check_script:NnTF \l_fontspec_font {latn}
          {
           \@@_warning:nx {script-not-exist-latn} {#1}
           \tl_set:Nn \l_fontspec_script_tl {latn}
           \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
          }
          {
           \@@_warning:nx {script-not-exist} {#1}
          }
        }
      }
    }
  }
```

12.4.2 Language

```
\keys_define:nn { fontspec-opentype } { Language .choice: }
\cs_new:Nn \fontspec_new_lang:nn
 {
  \keys_define:nn { fontspec-opentype } { Language / #1 .code:n =
  \@@_check_lang:NnTF \l_fontspec_font {#2}
     {
      \tl_set:Nn \l_fontspec_lang_tl {#2}
      \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
      \tl_gset:Nx \g_@@_single_feat_tl { language=#2 }
     }
     {
      \@@_warning:nx {language-not-exist} {#1}
      \keys_set:nn { fontspec-opentype } { Language = Default }
     }
  }
 }
```

Default

```
\@@_keys_define_code:nnn {fontspec-opentype}{ Language / Default }
```

```
 {
  \tl_set:Nn \l_fontspec_lang_tl {DFLT}
  \int_zero:N \l_@@_language_int
  \tl_gset:Nn \g_@@_single_feat_tl { language=DFLT }
 }
```

Turkish Turns out that many fonts use 'TUR' as their Turkish language tag rather than the specified 'TRK'. So we check for both:

```
\keys_define:nn {fontspec-opentype}
 {
  Language / Turkish .code:n =
   {
    \@@_check_lang:NnTF \l_fontspec_font {TRK}
     {
      \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
      \tl_set:Nn \l_fontspec_lang_tl {TRK}
      \tl_gset:Nn \g_@@_single_feat_tl { language=TRK }
     }
     {
      \@@_check_lang:NnTF \l_fontspec_font {TUR}
       {
        \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
        \tl_set:Nn \l_fontspec_lang_tl {TUR}
        \tl_gset:Nn \g_@@_single_feat_tl { language=TUR }
       }
       {
        \@@_warning:nx {language-not-exist} {Turkish}
        \keys_set:nn {fontspec-opentype} {Language=Default}
       }
     }
   }
 }
```

12.5 Backwards compatibility

Backwards compatibility:

```
\cs_new:Nn \@@_ot_compat:nn
  {
    \aliasfontfeatureoption {#1} {#2Off} {No#2}
  }
\@@_ot_compat:nn {Ligatures}   {Rare}
\@@_ot_compat:nn {Ligatures}   {Required}
\@@_ot_compat:nn {Ligatures}   {Common}
\@@_ot_compat:nn {Ligatures}   {Discretionary}
\@@_ot_compat:nn {Ligatures}   {Contextual}
\@@_ot_compat:nn {Ligatures}   {Historic}
\@@_ot_compat:nn {Numbers}     {SlashedZero}
\@@_ot_compat:nn {Contextuals} {Swash}
\@@_ot_compat:nn {Contextuals} {Alternate}
\@@_ot_compat:nn {Contextuals} {WordInitial}
```

```
\@@_ot_compat:nn {Contextuals} {WordFinal}
\@@_ot_compat:nn {Contextuals} {LineFinal}
\@@_ot_compat:nn {Contextuals} {Inner}
\@@_ot_compat:nn {Diacritics}  {MarkToBase}
\@@_ot_compat:nn {Diacritics}  {MarkToMark}
\@@_ot_compat:nn {Diacritics}  {AboveBase}
\@@_ot_compat:nn {Diacritics}  {BelowBase}
```

12.6 Font script definitions

```
\newfontscript{Adlam}{adlm}
\newfontscript{Ahom}{ahom}
\newfontscript{Anatolian~Hieroglyphs}{hluw}
\newfontscript{Arabic}{arab}
\newfontscript{Armenian}{armn}
\newfontscript{Avestan}{avst}
\newfontscript{Balinese}{bali}
\newfontscript{Bamum}{bamu}
\newfontscript{Bassa~Vah}{bass}
\newfontscript{Batak}{batk}
\newfontscript{Bengali}{bng2,beng}
\newfontscript{Bhaiksuki}{bhks}
\newfontscript{Bopomofo}{bopo}
\newfontscript{Brahmi}{brah}
\newfontscript{Braille}{brai}
\newfontscript{Buginese}{bugi}
\newfontscript{Buhid}{buhd}
\newfontscript{Byzantine~Music}{byzm}
\newfontscript{Canadian~Syllabics}{cans}
\newfontscript{Carian}{cari}
\newfontscript{Caucasian~Albanian}{aghb}
\newfontscript{Chakma}{cakm}
\newfontscript{Cham}{cham}
\newfontscript{Cherokee}{cher}
\newfontscript{CJK~Ideographic}{hani}
\newfontscript{Coptic}{copt}
\newfontscript{Cypriot~Syllabary}{cprt}
\newfontscript{Cyrillic}{cyrl}
\newfontscript{Default}{DFLT}
\newfontscript{Deseret}{dsrt}
\newfontscript{Devanagari}{dev2,deva}
\newfontscript{Duployan}{dupl}
\newfontscript{Egyptian~Hieroglyphs}{egyp}
\newfontscript{Elbasan}{elba}
\newfontscript{Ethiopic}{ethi}
\newfontscript{Georgian}{geor}
\newfontscript{Glagolitic}{glag}
\newfontscript{Gothic}{goth}
\newfontscript{Grantha}{gran}
\newfontscript{Greek}{grek}
\newfontscript{Gujarati}{gjr2,gujr}
```

```
\newfontscript{Gurmukhi}{gur2,guru}
\newfontscript{Hangul~Jamo}{jamo}
\newfontscript{Hangul}{hang}
\newfontscript{Hanunoo}{hano}
\newfontscript{Hatran}{hatr}
\newfontscript{Hebrew}{hebr}
\newfontscript{Hiragana~and~Katakana}{kana}
\newfontscript{Imperial~Aramaic}{armi}
\newfontscript{Inscriptional~Pahlavi}{phli}
\newfontscript{Inscriptional~Parthian}{prti}
\newfontscript{Javanese}{java}
\newfontscript{Kaithi}{kthi}
\newfontscript{Kannada}{knd2,knda}
\newfontscript{Kayah~Li}{kali}
\newfontscript{Kharosthi}{khar}
\newfontscript{Khmer}{khmr}
\newfontscript{Khojki}{khoj}
\newfontscript{Khudawadi}{sind}
\newfontscript{Lao}{lao~}
\newfontscript{Latin}{latn}
\newfontscript{Lepcha}{lepc}
\newfontscript{Limbu}{limb}
\newfontscript{Linear~A}{lina}
\newfontscript{Linear~B}{linb}
\newfontscript{Lisu}{lisu}
\newfontscript{Lycian}{lyci}
\newfontscript{Lydian}{lydi}
\newfontscript{Mahajani}{mahj}
\newfontscript{Malayalam}{mlm2,mlym}
\newfontscript{Mandaic}{mand}
\newfontscript{Manichaean}{mani}
\newfontscript{Marchen}{marc}
\newfontscript{Math}{math}
\newfontscript{Meitei~Mayek}{mtei}
\newfontscript{Mende~Kikakui}{mend}
\newfontscript{Meroitic~Cursive}{merc}
\newfontscript{Meroitic~Hieroglyphs}{mero}
\newfontscript{Miao}{plrd}
\newfontscript{Modi}{modi}
\newfontscript{Mongolian}{mong}
\newfontscript{Mro}{mroo}
\newfontscript{Multani}{mult}
\newfontscript{Musical~Symbols}{musc}
\newfontscript{Myanmar}{mym2,mymr}
\newfontscript{N'Ko}{nko~}
\newfontscript{Nabataean}{nbat}
\newfontscript{Newa}{newa}
\newfontscript{Odia}{ory2,orya}
\newfontscript{Ogham}{ogam}
\newfontscript{Ol~Chiki}{olck}
\newfontscript{Old~Italic}{ital}
```

```
\newfontscript{Old~Hungarian}{hung}
\newfontscript{Old~North~Arabian}{narb}
\newfontscript{Old~Permic}{perm}
\newfontscript{Old~Persian~Cuneiform}{xpeo}
\newfontscript{Old~South~Arabian}{sarb}
\newfontscript{Old~Turkic}{orkh}
\newfontscript{Osage}{osge}
\newfontscript{Osmanya}{osma}
\newfontscript{Pahawh~Hmong}{hmng}
\newfontscript{Palmyrene}{palm}
\newfontscript{Pau~Cin~Hau}{pauc}
\newfontscript{Phags-pa}{phag}
\newfontscript{Phoenician}{phnx}
\newfontscript{Psalter~Pahlavi}{phlp}
\newfontscript{Rejang}{rjng}
\newfontscript{Runic}{runr}
\newfontscript{Samaritan}{samr}
\newfontscript{Saurashtra}{saur}
\newfontscript{Sharada}{shrd}
\newfontscript{Shavian}{shaw}
\newfontscript{Siddham}{sidd}
\newfontscript{Sign~Writing}{sgnw}
\newfontscript{Sinhala}{sinh}
\newfontscript{Sora~Sompeng}{sora}
\newfontscript{Sumero-Akkadian~Cuneiform}{xsux}
\newfontscript{Sundanese}{sund}
\newfontscript{Syloti~Nagri}{sylo}
\newfontscript{Syriac}{syrc}
\newfontscript{Tagalog}{tglg}
\newfontscript{Tagbanwa}{tagb}
\newfontscript{Tai~Le}{tale}
\newfontscript{Tai~Lu}{talu}
\newfontscript{Tai~Tham}{lana}
\newfontscript{Tai~Viet}{tavt}
\newfontscript{Takri}{takr}
\newfontscript{Tamil}{tml2,taml}
\newfontscript{Tangut}{tang}
\newfontscript{Telugu}{tel2,telu}
\newfontscript{Thaana}{thaa}
\newfontscript{Thai}{thai}
\newfontscript{Tibetan}{tibt}
\newfontscript{Tifinagh}{tfng}
\newfontscript{Tirhuta}{tirh}
\newfontscript{Ugaritic~Cuneiform}{ugar}
\newfontscript{Vai}{vai~}
\newfontscript{Warang~Citi}{wara}
\newfontscript{Yi}{yi~~}
```

For convenience or backwards compatibility:

```
\newfontscript{CJK}{hani}
\newfontscript{Kana}{kana}
\newfontscript{Maths}{math}
```

```
\newfontscript{N'ko}{nko~}
\newfontscript{Oriya}{ory2,orya}
```

12.7 Font language definitions

```
\newfontlanguage{Abaza}{ABA}
\newfontlanguage{Abkhazian}{ABK}
\newfontlanguage{Adyghe}{ADY}
\newfontlanguage{Afrikaans}{AFK}
\newfontlanguage{Afar}{AFR}
\newfontlanguage{Agaw}{AGW}
\newfontlanguage{Altai}{ALT}
\newfontlanguage{Amharic}{AMH}
\newfontlanguage{Arabic}{ARA}
\newfontlanguage{Aari}{ARI}
\newfontlanguage{Arakanese}{ARK}
\newfontlanguage{Assamese}{ASM}
\newfontlanguage{Athapaskan}{ATH}
\newfontlanguage{Avar}{AVR}
\newfontlanguage{Awadhi}{AWA}
\newfontlanguage{Aymara}{AYM}
\newfontlanguage{Azeri}{AZE}
\newfontlanguage{Badaga}{BAD}
\newfontlanguage{Baghelkhandi}{BAG}
\newfontlanguage{Balkar}{BAL}
\newfontlanguage{Baule}{BAU}
\newfontlanguage{Berber}{BBR}
\newfontlanguage{Bench}{BCH}
\newfontlanguage{Bible~Cree}{BCR}
\newfontlanguage{Belarussian}{BEL}
\newfontlanguage{Bemba}{BEM}
\newfontlanguage{Bengali}{BEN}
\newfontlanguage{Bulgarian}{BGR}
\newfontlanguage{Bhili}{BHI}
\newfontlanguage{Bhojpuri}{BHO}
\newfontlanguage{Bikol}{BIK}
\newfontlanguage{Bilen}{BIL}
\newfontlanguage{Blackfoot}{BKF}
\newfontlanguage{Balochi}{BLI}
\newfontlanguage{Balante}{BLN}
\newfontlanguage{Balti}{BLT}
\newfontlanguage{Bambara}{BMB}
\newfontlanguage{Bamileke}{BML}
\newfontlanguage{Breton}{BRE}
\newfontlanguage{Brahui}{BRH}
\newfontlanguage{Braj~Bhasha}{BRI}
\newfontlanguage{Burmese}{BRM}
\newfontlanguage{Bashkir}{BSH}
\newfontlanguage{Beti}{BTI}
\newfontlanguage{Catalan}{CAT}
\newfontlanguage{Cebuano}{CEB}
\newfontlanguage{Chechen}{CHE}
```

```
\newfontlanguage{Chaha~Gurage}{CHG}
\newfontlanguage{Chattisgarhi}{CHH}
\newfontlanguage{Chichewa}{CHI}
\newfontlanguage{Chukchi}{CHK}
\newfontlanguage{Chipewyan}{CHP}
\newfontlanguage{Cherokee}{CHR}
\newfontlanguage{Chuvash}{CHU}
\newfontlanguage{Comorian}{CMR}
\newfontlanguage{Coptic}{COP}
\newfontlanguage{Cree}{CRE}
\newfontlanguage{Carrier}{CRR}
\newfontlanguage{Crimean~Tatar}{CRT}
\newfontlanguage{Church~Slavonic}{CSL}
\newfontlanguage{Czech}{CSY}
\newfontlanguage{Danish}{DAN}
\newfontlanguage{Dargwa}{DAR}
\newfontlanguage{Woods~Cree}{DCR}
\newfontlanguage{German}{DEU}
\newfontlanguage{Dogri}{DGR}
\newfontlanguage{Divehi}{DIV}
\newfontlanguage{Djerma}{DJR}
\newfontlanguage{Dangme}{DNG}
\newfontlanguage{Dinka}{DNK}
\newfontlanguage{Dungan}{DUN}
\newfontlanguage{Dzongkha}{DZN}
\newfontlanguage{Ebira}{EBI}
\newfontlanguage{Eastern~Cree}{ECR}
\newfontlanguage{Edo}{EDO}
\newfontlanguage{Efik}{EFI}
\newfontlanguage{Greek}{ELL}
\newfontlanguage{English}{ENG}
\newfontlanguage{Erzya}{ERZ}
\newfontlanguage{Spanish}{ESP}
\newfontlanguage{Estonian}{ETI}
\newfontlanguage{Basque}{EUQ}
\newfontlanguage{Evenki}{EVK}
\newfontlanguage{Even}{EVN}
\newfontlanguage{Ewe}{EWE}
\newfontlanguage{French~Antillean}{FAN}
\newfontlanguage{Farsi}{FAR}
\newfontlanguage{Parsi}{FAR}
\newfontlanguage{Persian}{FAR}
\newfontlanguage{Finnish}{FIN}
\newfontlanguage{Fijian}{FJI}
\newfontlanguage{Flemish}{FLE}
\newfontlanguage{Forest~Nenets}{FNE}
\newfontlanguage{Fon}{FON}
\newfontlanguage{Faroese}{FOS}
\newfontlanguage{French}{FRA}
\newfontlanguage{Frisian}{FRI}
\newfontlanguage{Friulian}{FRL}
```

```
\newfontlanguage{Futa}{FTA}
\newfontlanguage{Fulani}{FUL}
\newfontlanguage{Ga}{GAD}
\newfontlanguage{Gaelic}{GAE}
\newfontlanguage{Gagauz}{GAG}
\newfontlanguage{Galician}{GAL}
\newfontlanguage{Garshuni}{GAR}
\newfontlanguage{Garhwali}{GAW}
\newfontlanguage{Ge'ez}{GEZ}
\newfontlanguage{Gilyak}{GIL}
\newfontlanguage{Gumuz}{GMZ}
\newfontlanguage{Gondi}{GON}
\newfontlanguage{Greenlandic}{GRN}
\newfontlanguage{Garo}{GRO}
\newfontlanguage{Guarani}{GUA}
\newfontlanguage{Gujarati}{GUJ}
\newfontlanguage{Haitian}{HAI}
\newfontlanguage{Halam}{HAL}
\newfontlanguage{Harauti}{HAR}
\newfontlanguage{Hausa}{HAU}
\newfontlanguage{Hawaiin}{HAW}
\newfontlanguage{Hammer-Banna}{HBN}
\newfontlanguage{Hiligaynon}{HIL}
\newfontlanguage{Hindi}{HIN}
\newfontlanguage{High~Mari}{HMA}
\newfontlanguage{Hindko}{HND}
\newfontlanguage{Ho}{HO}
\newfontlanguage{Harari}{HRI}
\newfontlanguage{Croatian}{HRV}
\newfontlanguage{Hungarian}{HUN}
\newfontlanguage{Armenian}{HYE}
\newfontlanguage{Igbo}{IBO}
\newfontlanguage{Ijo}{IJO}
\newfontlanguage{Ilokano}{ILO}
\newfontlanguage{Indonesian}{IND}
\newfontlanguage{Ingush}{ING}
\newfontlanguage{Inuktitut}{INU}
\newfontlanguage{Irish}{IRI}
\newfontlanguage{Irish~Traditional}{IRT}
\newfontlanguage{Icelandic}{ISL}
\newfontlanguage{Inari~Sami}{ISM}
\newfontlanguage{Italian}{ITA}
\newfontlanguage{Hebrew}{IWR}
\newfontlanguage{Javanese}{JAV}
\newfontlanguage{Yiddish}{JII}
\newfontlanguage{Japanese}{JAN}
\newfontlanguage{Judezmo}{JUD}
\newfontlanguage{Jula}{JUL}
\newfontlanguage{Kabardian}{KAB}
\newfontlanguage{Kachchi}{KAC}
\newfontlanguage{Kalenjin}{KAL}
```

```
\newfontlanguage{Kannada}{KAN}
\newfontlanguage{Karachay}{KAR}
\newfontlanguage{Georgian}{KAT}
\newfontlanguage{Kazakh}{KAZ}
\newfontlanguage{Kebena}{KEB}
\newfontlanguage{Khutsuri~Georgian}{KGE}
\newfontlanguage{Khakass}{KHA}
\newfontlanguage{Khanty-Kazim}{KHK}
\newfontlanguage{Khmer}{KHM}
\newfontlanguage{Khanty-Shurishkar}{KHS}
\newfontlanguage{Khanty-Vakhi}{KHV}
\newfontlanguage{Khowar}{KHW}
\newfontlanguage{Kikuyu}{KIK}
\newfontlanguage{Kirghiz}{KIR}
\newfontlanguage{Kisii}{KIS}
\newfontlanguage{Kokni}{KKN}
\newfontlanguage{Kalmyk}{KLM}
\newfontlanguage{Kamba}{KMB}
\newfontlanguage{Kumaoni}{KMN}
\newfontlanguage{Komo}{KMO}
\newfontlanguage{Komso}{KMS}
\newfontlanguage{Kanuri}{KNR}
\newfontlanguage{Kodagu}{KOD}
\newfontlanguage{Korean~Old~Hangul}{KOH}
\newfontlanguage{Konkani}{KOK}
\newfontlanguage{Kikongo}{KON}
\newfontlanguage{Komi-Permyak}{KOP}
\newfontlanguage{Korean}{KOR}
\newfontlanguage{Komi-Zyrian}{KOZ}
\newfontlanguage{Kpelle}{KPL}
\newfontlanguage{Krio}{KRI}
\newfontlanguage{Karakalpak}{KRK}
\newfontlanguage{Karelian}{KRL}
\newfontlanguage{Karaim}{KRM}
\newfontlanguage{Karen}{KRN}
\newfontlanguage{Koorete}{KRT}
\newfontlanguage{Kashmiri}{KSH}
\newfontlanguage{Khasi}{KSI}
\newfontlanguage{Kildin~Sami}{KSM}
\newfontlanguage{Kui}{KUI}
\newfontlanguage{Kulvi}{KUL}
\newfontlanguage{Kumyk}{KUM}
\newfontlanguage{Kurdish}{KUR}
\newfontlanguage{Kurukh}{KUU}
\newfontlanguage{Kuy}{KUY}
\newfontlanguage{Koryak}{KYK}
\newfontlanguage{Ladin}{LAD}
\newfontlanguage{Lahuli}{LAH}
\newfontlanguage{Lak}{LAK}
\newfontlanguage{Lambani}{LAM}
\newfontlanguage{Lao}{LAO}
```

```
\newfontlanguage{Latin}{LAT}
\newfontlanguage{Laz}{LAZ}
\newfontlanguage{L-Cree}{LCR}
\newfontlanguage{Ladakhi}{LDK}
\newfontlanguage{Lezgi}{LEZ}
\newfontlanguage{Lingala}{LIN}
\newfontlanguage{Low~Mari}{LMA}
\newfontlanguage{Limbu}{LMB}
\newfontlanguage{Lomwe}{LMW}
\newfontlanguage{Lower~Sorbian}{LSB}
\newfontlanguage{Lule~Sami}{LSM}
\newfontlanguage{Lithuanian}{LTH}
\newfontlanguage{Luba}{LUB}
\newfontlanguage{Luganda}{LUG}
\newfontlanguage{Luhya}{LUH}
\newfontlanguage{Luo}{LUO}
\newfontlanguage{Latvian}{LVI}
\newfontlanguage{Majang}{MAJ}
\newfontlanguage{Makua}{MAK}
\newfontlanguage{Malayalam~Traditional}{MAL}
\newfontlanguage{Mansi}{MAN}
\newfontlanguage{Marathi}{MAR}
\newfontlanguage{Marwari}{MAW}
\newfontlanguage{Mbundu}{MBN}
\newfontlanguage{Manchu}{MCH}
\newfontlanguage{Moose~Cree}{MCR}
\newfontlanguage{Mende}{MDE}
\newfontlanguage{Me'en}{MEN}
\newfontlanguage{Mizo}{MIZ}
\newfontlanguage{Macedonian}{MKD}
\newfontlanguage{Male}{MLE}
\newfontlanguage{Malagasy}{MLG}
\newfontlanguage{Malinke}{MLN}
\newfontlanguage{Malayalam~Reformed}{MLR}
\newfontlanguage{Malay}{MLY}
\newfontlanguage{Mandinka}{MND}
\newfontlanguage{Mongolian}{MNG}
\newfontlanguage{Manipuri}{MNI}
\newfontlanguage{Maninka}{MNK}
\newfontlanguage{Manx~Gaelic}{MNX}
\newfontlanguage{Moksha}{MOK}
\newfontlanguage{Moldavian}{MOL}
\newfontlanguage{Mon}{MON}
\newfontlanguage{Moroccan}{MOR}
\newfontlanguage{Maori}{MRI}
\newfontlanguage{Maithili}{MTH}
\newfontlanguage{Maltese}{MTS}
\newfontlanguage{Mundari}{MUN}
\newfontlanguage{Naga-Assamese}{NAG}
\newfontlanguage{Nanai}{NAN}
\newfontlanguage{Naskapi}{NAS}
```

```
\newfontlanguage{N-Cree}{NCR}
\newfontlanguage{Ndebele}{NDB}
\newfontlanguage{Ndonga}{NDG}
\newfontlanguage{Nepali}{NEP}
\newfontlanguage{Newari}{NEW}
\newfontlanguage{Nagari}{NGR}
\newfontlanguage{Norway~House~Cree}{NHC}
\newfontlanguage{Nisi}{NIS}
\newfontlanguage{Niuean}{NIU}
\newfontlanguage{Nkole}{NKL}
\newfontlanguage{N'ko}{NKO}
\newfontlanguage{Dutch}{NLD}
\newfontlanguage{Nogai}{NOG}
\newfontlanguage{Norwegian}{NOR}
\newfontlanguage{Northern~Sami}{NSM}
\newfontlanguage{Northern~Tai}{NTA}
\newfontlanguage{Esperanto}{NTO}
\newfontlanguage{Nynorsk}{NYN}
\newfontlanguage{Oji-Cree}{OCR}
\newfontlanguage{Ojibway}{OJB}
\newfontlanguage{Oriya}{ORI}
\newfontlanguage{Oromo}{ORO}
\newfontlanguage{Ossetian}{OSS}
\newfontlanguage{Palestinian~Aramaic}{PAA}
\newfontlanguage{Pali}{PAL}
\newfontlanguage{Punjabi}{PAN}
\newfontlanguage{Palpa}{PAP}
\newfontlanguage{Pashto}{PAS}
\newfontlanguage{Polytonic~Greek}{PGR}
\newfontlanguage{Pilipino}{PIL}
\newfontlanguage{Palaung}{PLG}
\newfontlanguage{Polish}{PLK}
\newfontlanguage{Provencal}{PRO}
\newfontlanguage{Portuguese}{PTG}
\newfontlanguage{Chin}{QIN}
\newfontlanguage{Rajasthani}{RAJ}
\newfontlanguage{R-Cree}{RCR}
\newfontlanguage{Russian~Buriat}{RBU}
\newfontlanguage{Riang}{RIA}
\newfontlanguage{Rhaeto-Romanic}{RMS}
\newfontlanguage{Romanian}{ROM}
\newfontlanguage{Romany}{ROY}
\newfontlanguage{Rusyn}{RSY}
\newfontlanguage{Ruanda}{RUA}
\newfontlanguage{Russian}{RUS}
\newfontlanguage{Sadri}{SAD}
\newfontlanguage{Sanskrit}{SAN}
\newfontlanguage{Santali}{SAT}
\newfontlanguage{Sayisi}{SAY}
\newfontlanguage{Sekota}{SEK}
\newfontlanguage{Selkup}{SEL}
```

```
\newfontlanguage{Sango}{SGO}
\newfontlanguage{Shan}{SHN}
\newfontlanguage{Sibe}{SIB}
\newfontlanguage{Sidamo}{SID}
\newfontlanguage{Silte~Gurage}{SIG}
\newfontlanguage{Skolt~Sami}{SKS}
\newfontlanguage{Slovak}{SKY}
\newfontlanguage{Slavey}{SLA}
\newfontlanguage{Slovenian}{SLV}
\newfontlanguage{Somali}{SML}
\newfontlanguage{Samoan}{SMO}
\newfontlanguage{Sena}{SNA}
\newfontlanguage{Sindhi}{SND}
\newfontlanguage{Sinhalese}{SNH}
\newfontlanguage{Soninke}{SNK}
\newfontlanguage{Sodo~Gurage}{SOG}
\newfontlanguage{Sotho}{SOT}
\newfontlanguage{Albanian}{SQI}
\newfontlanguage{Serbian}{SRB}
\newfontlanguage{Saraiki}{SRK}
\newfontlanguage{Serer}{SRR}
\newfontlanguage{South~Slavey}{SSL}
\newfontlanguage{Southern~Sami}{SSM}
\newfontlanguage{Suri}{SUR}
\newfontlanguage{Svan}{SVA}
\newfontlanguage{Swedish}{SVE}
\newfontlanguage{Swadaya~Aramaic}{SWA}
\newfontlanguage{Swahili}{SWK}
\newfontlanguage{Swazi}{SWZ}
\newfontlanguage{Sutu}{SXT}
\newfontlanguage{Syriac}{SYR}
\newfontlanguage{Tabasaran}{TAB}
\newfontlanguage{Tajiki}{TAJ}
\newfontlanguage{Tamil}{TAM}
\newfontlanguage{Tatar}{TAT}
\newfontlanguage{TH-Cree}{TCR}
\newfontlanguage{Telugu}{TEL}
\newfontlanguage{Tongan}{TGN}
\newfontlanguage{Tigre}{TGR}
\newfontlanguage{Tigrinya}{TGY}
\newfontlanguage{Thai}{THA}
\newfontlanguage{Tahitian}{THT}
\newfontlanguage{Tibetan}{TIB}
\newfontlanguage{Turkmen}{TKM}
\newfontlanguage{Temne}{TMN}
\newfontlanguage{Tswana}{TNA}
\newfontlanguage{Tundra~Nenets}{TNE}
\newfontlanguage{Tonga}{TNG}
\newfontlanguage{Todo}{TOD}
\newfontlanguage{Tsonga}{TSG}
\newfontlanguage{Turoyo~Aramaic}{TUA}
```

```
\newfontlanguage{Tulu}{TUL}
\newfontlanguage{Tuvin}{TUV}
\newfontlanguage{Twi}{TWI}
\newfontlanguage{Udmurt}{UDM}
\newfontlanguage{Ukrainian}{UKR}
\newfontlanguage{Urdu}{URD}
\newfontlanguage{Upper~Sorbian}{USB}
\newfontlanguage{Uyghur}{UYG}
\newfontlanguage{Uzbek}{UZB}
\newfontlanguage{Venda}{VEN}
\newfontlanguage{Vietnamese}{VIT}
\newfontlanguage{Wa}{WA}
\newfontlanguage{Wagdi}{WAG}
\newfontlanguage{West-Cree}{WCR}
\newfontlanguage{Welsh}{WEL}
\newfontlanguage{Wolof}{WLF}
\newfontlanguage{Tai~Lue}{XBD}
\newfontlanguage{Xhosa}{XHS}
\newfontlanguage{Yakut}{YAK}
\newfontlanguage{Yoruba}{YBA}
\newfontlanguage{Y-Cree}{YCR}
\newfontlanguage{Yi~Classic}{YIC}
\newfontlanguage{Yi~Modern}{YIM}
\newfontlanguage{Chinese~Hong~Kong}{ZHH}
\newfontlanguage{Chinese~Phonetic}{ZHP}
\newfontlanguage{Chinese~Simplified}{ZHS}
\newfontlanguage{Chinese~Traditional}{ZHT}
\newfontlanguage{Zande}{ZND}
\newfontlanguage{Zulu}{ZUL}
```

12.8 AAT feature definitions

These are only defined for XeTeX.

12.8.1 Ligatures

```
\@@_define_aat_feature_group:n {Ligatures}
\@@_define_aat_feature:nnnn      {Ligatures} {Required} {1} {0}
\@@_define_aat_feature:nnnn      {Ligatures} {NoRequired} {1} {1}
\@@_define_aat_feature:nnnn      {Ligatures} {Common} {1} {2}
\@@_define_aat_feature:nnnn      {Ligatures} {NoCommon} {1} {3}
\@@_define_aat_feature:nnnn      {Ligatures} {Rare} {1} {4}
\@@_define_aat_feature:nnnn      {Ligatures} {NoRare} {1} {5}
\@@_define_aat_feature:nnnn      {Ligatures} {Discretionary} {1} {4}
\@@_define_aat_feature:nnnn      {Ligatures} {NoDiscretionary} {1} {5}
\@@_define_aat_feature:nnnn      {Ligatures} {Logos} {1} {6}
\@@_define_aat_feature:nnnn      {Ligatures} {NoLogos} {1} {7}
\@@_define_aat_feature:nnnn      {Ligatures} {Rebus} {1} {8}
\@@_define_aat_feature:nnnn      {Ligatures} {NoRebus} {1} {9}
\@@_define_aat_feature:nnnn      {Ligatures} {Diphthong} {1} {10}
\@@_define_aat_feature:nnnn      {Ligatures} {NoDiphthong} {1} {11}
\@@_define_aat_feature:nnnn      {Ligatures} {Squared} {1} {12}
```

```
\@@_define_aat_feature:nnnn      {Ligatures} {NoSquared} {1} {13}
\@@_define_aat_feature:nnnn      {Ligatures} {AbbrevSquared} {1} {14}
\@@_define_aat_feature:nnnn      {Ligatures} {NoAbbrevSquared} {1} {15}
\@@_define_aat_feature:nnnn      {Ligatures} {Icelandic} {1} {32}
\@@_define_aat_feature:nnnn      {Ligatures} {NoIcelandic} {1} {33}
```

Emulate CM extra ligatures.

```
\keys_define:nn {fontspec-aat}
 {
  Ligatures / TeX .code:n =
   {
     \tl_set:Nn \l_@@_mapping_tl { tex-text }
   }
 }
```

12.8.2 Letters

```
\@@_define_aat_feature_group:n {Letters}
\@@_define_aat_feature:nnnn      {Letters} {Normal} {3} {0}
\@@_define_aat_feature:nnnn      {Letters} {Uppercase} {3} {1}
\@@_define_aat_feature:nnnn      {Letters} {Lowercase} {3} {2}
\@@_define_aat_feature:nnnn      {Letters} {SmallCaps} {3} {3}
\@@_define_aat_feature:nnnn      {Letters} {InitialCaps} {3} {4}
```

12.8.3 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.

```
\@@_define_aat_feature_group:n {Numbers}
\@@_define_aat_feature:nnnn      {Numbers} {Monospaced} {6} {0}
\@@_define_aat_feature:nnnn      {Numbers} {Proportional} {6} {1}
\@@_define_aat_feature:nnnn      {Numbers} {Lowercase} {21} {0}
\@@_define_aat_feature:nnnn      {Numbers} {OldStyle} {21} {0}
\@@_define_aat_feature:nnnn      {Numbers} {Uppercase} {21} {1}
\@@_define_aat_feature:nnnn      {Numbers} {Lining} {21} {1}
\@@_define_aat_feature:nnnn      {Numbers} {SlashedZero} {14} {5}
\@@_define_aat_feature:nnnn      {Numbers} {NoSlashedZero} {14} {4}
```

12.8.4 Contextuals

```
\@@_define_aat_feature_group:n   {Contextuals}
\@@_define_aat_feature:nnnn      {Contextuals} {WordInitial} {8} {0}
\@@_define_aat_feature:nnnn      {Contextuals} {NoWordInitial} {8} {1}
\@@_define_aat_feature:nnnn      {Contextuals} {WordFinal} {8} {2}
\@@_define_aat_feature:nnnn      {Contextuals} {NoWordFinal} {8} {3}
\@@_define_aat_feature:nnnn      {Contextuals} {LineInitial} {8} {4}
\@@_define_aat_feature:nnnn      {Contextuals} {NoLineInitial} {8} {5}
\@@_define_aat_feature:nnnn      {Contextuals} {LineFinal} {8} {6}
\@@_define_aat_feature:nnnn      {Contextuals} {NoLineFinal} {8} {7}
\@@_define_aat_feature:nnnn      {Contextuals} {Inner} {8} {8}
\@@_define_aat_feature:nnnn      {Contextuals} {NoInner} {8} {9}
```

12.8.5 Diacritics

```
\@@_define_aat_feature_group:n {Diacritics}
\@@_define_aat_feature:nnnn      {Diacritics} {Show} {9} {0}
\@@_define_aat_feature:nnnn      {Diacritics} {Hide} {9} {1}
\@@_define_aat_feature:nnnn      {Diacritics} {Decompose} {9} {2}
```

12.8.6 Vertical position

```
\@@_define_aat_feature_group:n {VerticalPosition}
\@@_define_aat_feature:nnnn      {VerticalPosition} {Normal} {10} {0}
\@@_define_aat_feature:nnnn      {VerticalPosition} {Superior} {10} {1}
\@@_define_aat_feature:nnnn      {VerticalPosition} {Inferior} {10} {2}
\@@_define_aat_feature:nnnn      {VerticalPosition} {Ordinal} {10} {3}
```

12.8.7 Fractions

```
\@@_define_aat_feature_group:n {Fractions}
\@@_define_aat_feature:nnnn      {Fractions} {On} {11} {1}
\@@_define_aat_feature:nnnn      {Fractions} {Off} {11} {0}
\@@_define_aat_feature:nnnn      {Fractions} {Diagonal} {11} {2}
```

12.8.8 Alternate

```
\@@_define_aat_feature_group:n  { Alternate }
\keys_define:nn {fontspec-aat}
 {
  Alternate .default:n = {0} ,
  Alternate / unknown .code:n =
   {
    \clist_map_inline:nn {#1}
      {
        \@@_make_AAT_feature:nn {17}{##1}
      }
   }
 }
```

12.8.9 Variant / StylisticSet

```
\@@_define_aat_feature_group:n  {Variant}
\keys_define:nn {fontspec-aat}
 {
  Variant .default:n = {0} ,
  Variant / unknown .code:n =
   {
    \clist_map_inline:nn {#1}
      { \@@_make_AAT_feature:nn {18}{##1} }
   }
 }
\aliasfontfeature{Variant}{StylisticSet}
```

```
\@@_define_aat_feature_group:n  {Vertical}
\keys_define:nn {fontspec-aat}
 {
  Vertical .choice: ,
  Vertical / RotatedGlyphs .code:n =
```

```
    {
      \__fontspec_update_featstr:n {vertical}
    }
}
```

12.8.10 Style

```
\@@_define_aat_feature_group:n {Style}
\@@_define_aat_feature:nnnn      {Style} {Italic} {32} {2}
\@@_define_aat_feature:nnnn      {Style} {Ruby} {28} {2}
\@@_define_aat_feature:nnnn      {Style} {Display} {19} {1}
\@@_define_aat_feature:nnnn      {Style} {Engraved} {19} {2}
\@@_define_aat_feature:nnnn      {Style} {TitlingCaps} {19} {4}
\@@_define_aat_feature:nnnn      {Style} {TallCaps} {19} {5}
```

12.8.11 CJK shape

```
\@@_define_aat_feature_group:n {CJKShape}
\@@_define_aat_feature:nnnn      {CJKShape} {Traditional} {20} {0}
\@@_define_aat_feature:nnnn      {CJKShape} {Simplified} {20} {1}
\@@_define_aat_feature:nnnn      {CJKShape} {JIS1978} {20} {2}
\@@_define_aat_feature:nnnn      {CJKShape} {JIS1983} {20} {3}
\@@_define_aat_feature:nnnn      {CJKShape} {JIS1990} {20} {4}
\@@_define_aat_feature:nnnn      {CJKShape} {Expert} {20} {10}
\@@_define_aat_feature:nnnn      {CJKShape} {NLC} {20} {13}
```

12.8.12 Character width

```
\@@_define_aat_feature_group:n {CharacterWidth}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Proportional} {22} {0}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Full} {22} {1}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Half} {22} {2}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Third} {22} {3}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Quarter} {22} {4}
\@@_define_aat_feature:nnnn      {CharacterWidth} {AlternateProportional} {22} {5}
\@@_define_aat_feature:nnnn      {CharacterWidth} {AlternateHalf} {22} {6}
\@@_define_aat_feature:nnnn      {CharacterWidth} {Default} {22} {7}
```

12.8.13 Annotation

```
\@@_define_aat_feature_group:n {Annotation}
\@@_define_aat_feature:nnnn      {Annotation} {Off} {24} {0}
\@@_define_aat_feature:nnnn      {Annotation} {Box} {24} {1}
\@@_define_aat_feature:nnnn      {Annotation} {RoundedBox} {24} {2}
\@@_define_aat_feature:nnnn      {Annotation} {Circle} {24} {3}
\@@_define_aat_feature:nnnn      {Annotation} {BlackCircle} {24} {4}
\@@_define_aat_feature:nnnn      {Annotation} {Parenthesis} {24} {5}
\@@_define_aat_feature:nnnn      {Annotation} {Period} {24} {6}
\@@_define_aat_feature:nnnn      {Annotation} {RomanNumerals} {24} {7}
\@@_define_aat_feature:nnnn      {Annotation} {Diamond} {24} {8}
\@@_define_aat_feature:nnnn      {Annotation} {BlackSquare} {24} {9}
\@@_define_aat_feature:nnnn      {Annotation} {BlackRoundSquare} {24} {10}
\@@_define_aat_feature:nnnn      {Annotation} {DoubleCircle} {24} {11}
```

13 Extended font encodings

To be removed after the 2017 release of LaTeX2e:

```
\providecommand\UnicodeFontFile[2]{"[#1]:#2"}
\providecommand\UnicodeFontName[2]{"#1:#2"}
⟨xetexx⟩\providecommand\UnicodeFontTeXLigatures{mapping=tex-text;}
⟨luatex⟩\providecommand\UnicodeFontTeXLigatures{+tlig;}

\providecommand\add@unicode@accent[2]{#2\char#1\relax}
\providecommand\DeclareUnicodeAccent[3]{%
  \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
}
```

\EncodingCommand

```
\DeclareDocumentCommand \EncodingCommand {mO{}m}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingCommand }
    \DeclareTextCommand{#1}{\UnicodeEncodingName}[#2]{#3}
  }
```

(*End definition for* `\EncodingCommand`*. This function is documented on page* ??.)

\EncodingAccent

```
\DeclareDocumentCommand \EncodingAccent {mm}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingAccent }
    \DeclareTextCommand{#1}{\UnicodeEncodingName}{\add@unicode@accent{#2}}
  }
```

(*End definition for* `\EncodingAccent`*. This function is documented on page* ??.)

\EncodingSymbol

```
\DeclareDocumentCommand \EncodingSymbol {mm}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingSymbol }
    \DeclareTextSymbol{#1}{\UnicodeEncodingName}{#2}
  }
```

(*End definition for* `\EncodingSymbol`*. This function is documented on page* ??.)

\EncodingComposite

```
\DeclareDocumentCommand \EncodingComposite {mmm}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingComposite }
    \DeclareTextComposite{#1}{\UnicodeEncodingName}{#2}{#3}
  }
```

(*End definition for* `\EncodingComposite`*. This function is documented on page* ??.)

\EncodingCompositeCommand

```
\DeclareDocumentCommand \EncodingCompositeCommand {mmm}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingCompositeCommand }
    \DeclareTextCompositeCommand{#1}{\UnicodeEncodingName}{#2}{#3}
  }
```

(*End definition for* `\EncodingCompositeCommand`*. This function is documented on page* ??.)

\DeclareUnicodeEncoding

```
\DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
  {
    \DeclareFontEncoding{#1}{}{}
    \DeclareErrorFont{#1}{lmr}{m}{n}{10}
    \DeclareFontSubstitution{#1}{lmr}{m}{n}
    \DeclareFontFamily{#1}{lmr}{}

    \DeclareFontShape{#1}{lmr}{m}{n}
      {<->\UnicodeFontFile{lmroman10-regular}{\UnicodeFontTeXLigatures}}{}
    \DeclareFontShape{#1}{lmr}{m}{it}
      {<->\UnicodeFontFile{lmroman10-italic}{\UnicodeFontTeXLigatures}}{}
    \DeclareFontShape{#1}{lmr}{m}{sc}
      {<->\UnicodeFontFile{lmromancaps10-regular}{\UnicodeFontTeXLigatures}}{}
    \DeclareFontShape{#1}{lmr}{bx}{n}
      {<->\UnicodeFontFile{lmroman10-bold}{\UnicodeFontTeXLigatures}}{}
    \DeclareFontShape{#1}{lmr}{bx}{it}
      {<->\UnicodeFontFile{lmroman10-bolditalic}{\UnicodeFontTeXLigatures}}{}

    \tl_set_eq:NN \l_@@_prev_unicode_name_tl \UnicodeEncodingName
    \tl_set:Nn \UnicodeEncodingName {#1}
    \bool_set_true:N \l_@@_defining_encoding_bool
    #2
    \bool_set_false:N \l_@@_defining_encoding_bool
    \tl_set_eq:NN \UnicodeEncodingName \l_@@_prev_unicode_name_tl
  }
```

(*End definition for* `\DeclareUnicodeEncoding`*. This function is documented on page* ??.)

\UndeclareSymbol
\UndeclareAccent
\UndeclareCommand

Synonyms for each other but all included for completeness.

```
\DeclareDocumentCommand \UndeclareSymbol {m}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \UndeclareSymbol }
    \UndeclareTextCommand {#1} {\UnicodeEncodingName}
  }
\DeclareDocumentCommand \UndeclareAccent {m}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \UndeclareAccent }
    \UndeclareTextCommand {#1} {\UnicodeEncodingName}
  }
```

```
\DeclareDocumentCommand \UndeclareCommand {m}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \UndeclareCommand }
    \UndeclareTextCommand {#1} {\UnicodeEncodingName}
  }
```

(*End definition for* `\UndeclareSymbol`, `\UndeclareAccent`, *and* `\UndeclareCommand`. *These functions are documented on page* ??.)

`\UndeclareComposite`

```
\DeclareDocumentCommand \UndeclareComposite {mm}
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \UndeclareComposite }
    \cs_undefine:c
      { \c_backslash_str \UnicodeEncodingName \token_to_str:N #1 - \tl_to_str:n {#2} }
  }
```

(*End definition for* `\UndeclareComposite`. *This function is documented on page* ??.)

14 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_@@_pkg_euler_loaded_bool
 }
 {
  \bool_set_false:N \g_@@_pkg_euler_loaded_bool
 }
\cs_set:Nn \fontspec_setup_maths:
 {
  \@ifpackageloaded{euler}
   {
    \bool_if:NTF \g_@@_pkg_euler_loaded_bool
     { \bool_set_true:N \g_@@_math_euler_bool }
     { \@@_error:n {euler-too-late} }
   }
   {}
  \@ifpackageloaded{lucbmath}{\bool_set_true:N \g_@@_math_lucida_bool}{}
  \@ifpackageloaded{lucidabr}{\bool_set_true:N \g_@@_math_lucida_bool}{}
  \@ifpackageloaded{lucimatx}{\bool_set_true:N \g_@@_math_lucida_bool}{}
```

Knuth's CM fonts fonts are all squashed together, combining letters, accents, text symbols and maths symbols all in the one font, `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 `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 `\hat` accent in `EulerFractur`, but it's *ugly*. So I ignore it. Sorry if this causes inconvenience.)

```
\DeclareSymbolFont{legacymaths}{OT1}{cmr}{m}{n}
\SetSymbolFont{legacymaths}{bold}{OT1}{cmr}{bx}{n}
\DeclareMathAccent{\acute}   {\mathalpha}{legacymaths}{19}
\DeclareMathAccent{\grave}   {\mathalpha}{legacymaths}{18}
\DeclareMathAccent{\ddot}    {\mathalpha}{legacymaths}{127}
\DeclareMathAccent{\tilde}   {\mathalpha}{legacymaths}{126}
\DeclareMathAccent{\bar}     {\mathalpha}{legacymaths}{22}
\DeclareMathAccent{\breve}   {\mathalpha}{legacymaths}{21}
\DeclareMathAccent{\check}   {\mathalpha}{legacymaths}{20}
\DeclareMathAccent{\hat}     {\mathalpha}{legacymaths}{94} % too bad, euler
\DeclareMathAccent{\dot}     {\mathalpha}{legacymaths}{95}
\DeclareMathAccent{\mathring}{\mathalpha}{legacymaths}{23}
```

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

```
% % fontmath.ltx:
% \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
% \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
%
% % amsmath.sty:
% \renewcommand{\colon}{\nobreak\mskip2mu\mathpunct{}\nonscript
%  \mkern-\thinmuskip{:}\mskip6muplus1mu\relax}
%
% % euler.sty:
% \DeclareMathSymbol{:}\mathrel  {EulerFraktur}{"3A}
%
% % lucbmath.sty:
% \DeclareMathSymbol{\@tempb}{\mathpunct}{operators}{58}
% \ifx\colon\@tempb
%   \DeclareMathSymbol{\colon}{\mathpunct}{operators}{58}
% \fi
% \DeclareMathSymbol{:}{\mathrel}{operators}{58}
```

($3A_{16} = 58_{10}$) 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.

```
\group_begin:
  \mathchardef\@tempa="6003A \relax
  \ifx\colon\@tempa
    \DeclareMathSymbol{\colon}{\mathpunct}{legacymaths}{58}
```

```
    \fi
  \group_end:
```

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

```
  \bool_if:NF \g_@@_math_euler_bool
   {
    \DeclareMathSymbol{!}{\mathclose}{legacymaths}{33}
    \DeclareMathSymbol{:}{\mathrel}  {legacymaths}{58}
    \DeclareMathSymbol{;}{\mathpunct}{legacymaths}{59}
    \DeclareMathSymbol{?}{\mathclose}{legacymaths}{63}
```

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.

```
    \bool_if:NF \g_@@_math_lucida_bool
     {
      \DeclareMathSymbol{0}{\mathalpha}{legacymaths}{`0}
      \DeclareMathSymbol{1}{\mathalpha}{legacymaths}{`1}
      \DeclareMathSymbol{2}{\mathalpha}{legacymaths}{`2}
      \DeclareMathSymbol{3}{\mathalpha}{legacymaths}{`3}
      \DeclareMathSymbol{4}{\mathalpha}{legacymaths}{`4}
      \DeclareMathSymbol{5}{\mathalpha}{legacymaths}{`5}
      \DeclareMathSymbol{6}{\mathalpha}{legacymaths}{`6}
      \DeclareMathSymbol{7}{\mathalpha}{legacymaths}{`7}
      \DeclareMathSymbol{8}{\mathalpha}{legacymaths}{`8}
      \DeclareMathSymbol{9}{\mathalpha}{legacymaths}{`9}
      \DeclareMathSymbol{\Gamma}{\mathalpha}{legacymaths}{0}
      \DeclareMathSymbol{\Delta}{\mathalpha}{legacymaths}{1}
      \DeclareMathSymbol{\Theta}{\mathalpha}{legacymaths}{2}
      \DeclareMathSymbol{\Lambda}{\mathalpha}{legacymaths}{3}
      \DeclareMathSymbol{\Xi}{\mathalpha}{legacymaths}{4}
      \DeclareMathSymbol{\Pi}{\mathalpha}{legacymaths}{5}
      \DeclareMathSymbol{\Sigma}{\mathalpha}{legacymaths}{6}
      \DeclareMathSymbol{\Upsilon}{\mathalpha}{legacymaths}{7}
      \DeclareMathSymbol{\Phi}{\mathalpha}{legacymaths}{8}
      \DeclareMathSymbol{\Psi}{\mathalpha}{legacymaths}{9}
      \DeclareMathSymbol{\Omega}{\mathalpha}{legacymaths}{10}
      \DeclareMathSymbol{+}{\mathbin}{legacymaths}{43}
      \DeclareMathSymbol{=}{\mathrel}{legacymaths}{61}
      \DeclareMathDelimiter{(}{\mathopen} {legacymaths}{40}{largesymbols}{0}
      \DeclareMathDelimiter{)}{\mathclose}{legacymaths}{41}{largesymbols}{1}
      \DeclareMathDelimiter{[}{\mathopen} {legacymaths}{91}{largesymbols}{2}
      \DeclareMathDelimiter{]}{\mathclose}{legacymaths}{93}{largesymbols}{3}
      \DeclareMathDelimiter{/}{\mathord}{legacymaths}{47}{largesymbols}{14}
      \DeclareMathSymbol{\mathdollar}{\mathord}{legacymaths}{36}
     }
   }
```

Finally, we change the font definitions for `\mathrm` and so on. These are defined using the `\g_@@_mathrm_tl` (…) macros, which default to `\rmdefault` but may be specified with the `\setmathrm` (…) commands in the preamble.

Since LaTeX only generally defines one level of boldness, we omit `\mathbf` in the bold maths series. It can be specified as per usual with `\setboldmathrm`, which stores the appropriate family name in `\g_@@_bfmathrm_tl`.

```
\DeclareSymbolFont{operators}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\updefault
\SetSymbolFont{operators}{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\updefault
\DeclareSymbolFontAlphabet\mathrm{operators}
\SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\itdefault
\SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\updefault
\SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g_@@_mathsf_tl\mddefault\updefault
\SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g_@@_mathtt_tl\mddefault\updefault
\SetSymbolFont{operators}{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\updefault
\tl_if_empty:NTF \g_@@_bfmathrm_tl
 {
  \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\itdefault
 }
 {
  \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\updefault
  \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\bfdefault\updefault
  \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\itdefault
 }
\SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\updefault
\SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\updefault
}
```

(*End definition for* `\fontspec_setup_maths:`*. This function is documented on page* ??.)

`\fontspec_maybe_setup_maths:`

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 'LaTeXFont Catalogue': 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 `\unless\ifnum\Gamma=28672\relax\bool_set_false:N \g_@@_math_bool\fi` be a better test? This needs more cooperation with euler and lucida, I think.

```
\cs_new:Nn \fontspec_maybe_setup_maths:
 {
  \@ifpackageloaded{anttor}
   {
    \ifx\define@antt@mathversions a\bool_set_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{arevmath}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{eulervm}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathdesign}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{concmath}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{cmbright}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathesf}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{gfsartemisia}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{gfsneohellenic}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{iwona}
   {
    \ifx\define@iwona@mathversions a\bool_set_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{kpfonts}{\bool_set_false:N \g_@@_math_bool}{}
```

```
  \@ifpackageloaded{kmath}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{kurier}
   {
    \ifx\define@kurier@mathversions a\bool_set_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{fouriernc}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{fourier}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{lmodern}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathpazo}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathptmx}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{MinionPro}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{unicode-math}{\bool_set_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{breqn}{\bool_set_false:N \g_@@_math_bool}{}
  \bool_if:NT \g_@@_math_bool
   {
    \@@_info:n {setup-math}
    \fontspec_setup_maths:
   }
 }
\AtBeginDocument{\fontspec_maybe_setup_maths:}
```

(*End definition for* `\fontspec_maybe_setup_maths:`*. This function is documented on page* ??.)

15 Closing code

15.1 Finishing up

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

```
\bool_if:NT \g_@@_cfg_bool
 {
  \InputIfFileExists{fontspec.cfg}
    {}
    {\typeout{No~ fontspec.cfg~ file~ found;~ no~ configuration~ loaded.}}
 }
```

16 Changes to the NFSS

⟨*fontspec⟩

16.1 Italic small caps and so on

`\sishape` `\textsi`

These commands for actually selecting italic small caps have been defined for many years; I'm inclined to drop them. They're probably used very infrequently; I personally prefer just writing `\textit{\textsc{...}}` instead.

```
\providecommand*\itscdefault{\itdefault\scdefault}
\providecommand*\slscdefault{\sldefault\scdefault}
\DeclareRobustCommand{\sishape}
 {
  \not@math@alphabet\sishape\relax
  \fontshape{\itscdefault}\selectfont
```

```
  }
\DeclareTextFontCommand{\textsi}{\sishape}
```

(*End definition for* `\sishape` *and* `\textsi`*. These functions are documented on page* ??.)

LaTeX's 'shape' font axis needs to be overloaded to support italic small caps and slanted small caps. These are the combinations to support:

```
\cs_new:Nn \@@_shape_merge:nn { c_@@_shape_#1_#2_tl }
\tl_const:cn { \@@_shape_merge:nn \itdefault   \scdefault } {\itscdefault}
\tl_const:cn { \@@_shape_merge:nn \sldefault   \scdefault } {\slscdefault}
\tl_const:cn { \@@_shape_merge:nn \scdefault   \itdefault } {\itscdefault}
\tl_const:cn { \@@_shape_merge:nn \scdefault   \sldefault } {\slscdefault}
\tl_const:cn { \@@_shape_merge:nn \slscdefault \itdefault } {\itscdefault}
\tl_const:cn { \@@_shape_merge:nn \itscdefault \sldefault } {\slscdefault}
\tl_const:cn { \@@_shape_merge:nn \itscdefault \updefault } {\scdefault}
\tl_const:cn { \@@_shape_merge:nn \slscdefault \updefault } {\scdefault}
```

`\fontspec_merge_shape:n` These macros enable the overload on the `\..shape` commands. First, a shape 'new+current' (prefix) or 'current+new' (suffix) is tried. If not found, fall back on the 'new' shape.

```
\cs_new:Nn \fontspec_merge_shape:n
  {
    \@@_if_merge_shape:nTF {#1}
      { \fontshape { \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} } } \selectfont }
      { \fontshape {#1} \selectfont }
  }
```

The following is rather specific; it only returns true if the merged shape exists, but more importantly also if the merged shape is defined for the current font.

```
\prg_new_conditional:Nnn \@@_if_merge_shape:n {TF}
  {
    \bool_lazy_and:nnTF
      { \tl_if_exist_p:c { \@@_shape_merge:nn {\f@shape} {#1} } }
      {
        \cs_if_exist_p:c
          {
            \f@encoding/\f@family/\f@series/
            \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} }
          }
      }
    \prg_return_true: \prg_return_false:
  }
```

(*End definition for* `\fontspec_merge_shape:n`*. This function is documented on page* ??.)

`\itshape` `\scshape` `\upshape` `\slshape` The original `\..shape` commands are redefined to use the merge shape macro.

```
\DeclareRobustCommand \itshape
 {
  \not@math@alphabet\itshape\mathit
  \fontspec_merge_shape:n\itdefault
 }
\DeclareRobustCommand \slshape
 {
  \not@math@alphabet\slshape\relax
```

```
  \fontspec_merge_shape:n\sldefault
 }
\DeclareRobustCommand \scshape
 {
  \not@math@alphabet\scshape\relax
  \fontspec_merge_shape:n\scdefault
 }
\DeclareRobustCommand \upshape
 {
  \not@math@alphabet\upshape\relax
  \fontspec_merge_shape:n\updefault
 }
```

(*End definition for* `\itshape` *and others. These functions are documented on page* ??.)

16.2 Emphasis

`\emfontdeclare`

```
\cs_new_protected:Npn \emfontdeclare #1
  {
    \prop_clear:N    \g_@@_em_prop
    \int_zero:N      \l_@@_emdef_int
    \bool_set_true:N \g_@@_em_normalise_slant_bool

    \tl_if_in:nnT {#1} {\slshape}
      {
        \tl_if_in:nnT {#1} {\itshape}
          {
            \bool_set_false:N \g_@@_em_normalise_slant_bool
          }
      }

    \group_begin:
      \normalfont
      \clist_map_inline:nn {\emreset,#1}
        {
          ##1
          \prop_gput_if_new:NxV \g_@@_em_prop { \f@shape } { \l_@@_emdef_int }
          \prop_gput:Nxn \g_@@_em_prop { switch-\int_use:N \l_@@_emdef_int } { ##1 }
          \int_incr:N \l_@@_emdef_int
        }
    \group_end:
  }
```

(*End definition for* `\emfontdeclare`*. This function is documented on page* ??.)

`\em`

```
\DeclareRobustCommand \em
  {
    \@nomath\em
    \tl_set:Nx \l_@@_emshape_query_tl { \f@shape }

```

```
    \bool_if:NT \g_@@_em_normalise_slant_bool
      {
        \tl_replace_all:Nnn \l_@@_emshape_query_tl {/sl} {/it}
      }

⟨debug⟩ \typeout{Emph~ level:~\int_use:N \l_@@_em_int}
    \prop_get:NxNT \g_@@_em_prop { \l_@@_emshape_query_tl } \l_@@_em_tmp_tl
      {
        \int_set:Nn \l_@@_em_int { \l_@@_em_tmp_tl }
⟨debug⟩ \typeout{Shape~ (\l_@@_emshape_query_tl)~ detected;~ new~ level:~\int_use:N \l_@@_em_
      }

    \int_incr:N \l_@@_em_int

    \prop_get:NxNTF \g_@@_em_prop { switch-\int_use:N \l_@@_em_int } \l_@@_em_switch_tl
      { \l_@@_em_switch_tl }
      {
        \int_zero:N \l_@@_em_int
        \emreset
      }

  }
```

(*End definition for* `\em`*. This function is documented on page* ??.)

`\emph` `\emshape` `\eminnershape` `\emreset`

```
\DeclareTextFontCommand{\emph}{\em}
\cs_set:Npn \emreset { \upshape }
\cs_set:Npn \emshape { \itshape }
\cs_set:Npn \eminnershape { \upshape }
```

(*End definition for* `\emph` *and others. These functions are documented on page* ??.)

16.3 Strong emphasis

`\strongfontdeclare`

```
\cs_new_protected:Npn \strongfontdeclare #1
  {
    \prop_clear:N    \g_@@_strong_prop
    \int_zero:N      \l_@@_strongdef_int

    \group_begin:
      \normalfont
      \clist_map_inline:nn {\strongreset,#1}
        {
          ##1
          \prop_gput_if_new:NxV \g_@@_strong_prop { \f@series } { \l_@@_strongdef_int }
          \prop_gput:Nxn \g_@@_strong_prop { switch-\int_use:N \l_@@_strongdef_int } { ##1 }
          \int_incr:N \l_@@_strongdef_int
        }
    \group_end:
  }
```

(*End definition for* `\strongfontdeclare`*. This function is documented on page* ??.)

`\strongenv`

```
\DeclareRobustCommand \strongenv
  {
    \@nomath\strongenv

⟨debug⟩ \typeout{Strong~ level:~\int_use:N \l_@@_strong_int}
    \prop_get:NxNT \g_@@_strong_prop { \f@series } \l_@@_strong_tmp_tl
      {
        \int_set:Nn \l_@@_strong_int { \l_@@_strong_tmp_tl }
⟨debug⟩ \typeout{Series~ (\f@series)~ detected;~ new~ level:~\int_use:N \l_@@_strong_int}
      }

    \int_incr:N \l_@@_strong_int

    \prop_get:NxNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_sw
      { \l_@@_strong_switch_tl }
      {
        \int_zero:N \l_@@_strong_int
        \strongreset
      }

  }
```

(*End definition for* `\strongenv`*. This function is documented on page* ??.)

`\strong`
`\strongreset`

```
\DeclareTextFontCommand{\strong}{\strongenv}
\cs_set:Npn \strongreset {}
```

(*End definition for* `\strong` *and* `\strongreset`*. These functions are documented on page* ??.)

`\reset@font` Ensure nesting resets when necessary:

```
\cs_set:Npn \reset@font
  {
    \normalfont
    \int_zero:N \l_@@_em_int
    \int_zero:N \l_@@_strong_int
  }
```

(*End definition for* `\reset@font`*. This function is documented on page* ??.)

Programmer's interface for setting nesting levels:

```
\cs_new:Nn \fontspec_set_em_level:n     { \int_set:Nn \l_@@_em_int     {#1} }
\cs_new:Nn \fontspec_set_strong_level:n { \int_set:Nn \l_@@_strong_int {#1} }
```

Defaults:

```
\strongfontdeclare{ \bfseries }
\emfontdeclare{ \emshape, \eminnershape }
```

⟨/fontspec⟩

17 Patching code

⟨*fontspec⟩

17.1 \-

\- This macro is courtesy of Frank Mittelbach and the LaTeX 2_ε source code.

```
\DeclareRobustCommand{\-}
 {
  \discretionary
   {
    \char\ifnum\hyphenchar\font<\z@
           \xlx@defaulthyphenchar
         \else
           \hyphenchar\font
         \fi
   }{}{}
 }
\def\xlx@defaulthyphenchar{`\-}
```

(*End definition for* `\-`. *This function is documented on page* ??.)

17.2 Verbatims

Many thanks to Apostolos Syropoulos for discovering this problem and writing the redefinion of LaTeX's `verbatim` environment and `\verb*` command.

\fontspec_visible_space: Print U+2423: OPEN BOX, which is used to visibly display a space character.

```
\cs_new:Nn \fontspec_visible_space:
 {
  \@@_primitive_font_glyph_if_exist:NnTF \font {"2423}
   { \char"2423\scan_stop: }
   { \fontspec_visible_space_fallback: }
 }
```

(*End definition for* `\fontspec_visible_space:`. *This function is documented on page* ??.)

\fontspec_visible_space_fallback: If the current font doesn't have U+2423: OPEN BOX, use Latin Modern Mono instead.

```
\cs_new:Nn \fontspec_visible_space_fallback:
 {
  {
   \usefont{\g_fontspec_encoding_tl}{lmtt}{\f@series}{\f@shape}
   \textvisiblespace
  }
 }
```

(*End definition for* `\fontspec_visible_space_fallback:`. *This function is documented on page* ??.)

\fontspec_print_visible_spaces: Helper macro to turn spaces (`^^20`) active and print visible space instead.

```
\group_begin:
\char_set_catcode_active:n{"20}%
\cs_gset:Npn\fontspec_print_visible_spaces:{%
\char_set_catcode_active:n{"20}%
```

```
\cs_set_eq:NN^^20\fontspec_visible_space:%
}%
\group_end:
```

(*End definition for* `\fontspec_print_visible_spaces:`*. This function is documented on page* ??.)

`\verb` `\verb*` Redefine `\verb` to use `\fontspec_print_visible_spaces:`.

```
\def\verb
 {
  \relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup
    \verb@eol@error \let\do\@makeother \dospecials
    \verbatim@font\@noligs
    \@ifstar\@@sverb\@verb
 }
\def\@@sverb{\fontspec_print_visible_spaces:\@sverb}
```

(*End definition for* `\verb` *and* `\verb*`*. These functions are documented on page* ??.)

It's better to put small things into `\AtBeginDocument`, so here we go:

```
\AtBeginDocument
 {
  \fontspec_patch_verbatim:
  \fontspec_patch_moreverb:
  \fontspec_patch_fancyvrb:
  \fontspec_patch_listings:
 }
```

`verbatim*` With the verbatim package.

```
\cs_set:Npn \fontspec_patch_verbatim:
 {
  \@ifpackageloaded{verbatim}
   {
    \cs_set:cpn {verbatim*}
     {
      \group_begin: \@verbatim \fontspec_print_visible_spaces: \verbatim@start
     }
   }
```

This is for vanilla LaTeX.

```
   {
    \cs_set:cpn {verbatim*}
     {
      \@verbatim \fontspec_print_visible_spaces: \@sxverbatim
     }
   }
 }
```

`listingcont*` This is for moreverb. The main `listing*` environment inherits this definition.

```
\cs_set:Npn \fontspec_patch_moreverb:
 {
  \@ifpackageloaded{moreverb}{
    \cs_set:cpn {listingcont*}
```

```
    {
     \cs_set:Npn \verbatim@processline
      {
       \thelisting@line \global\advance\listing@line\c_one
       \the\verbatim@line\par
      }
     \@verbatim \fontspec_print_visible_spaces: \verbatim@start
    }
  }{}
 }
```

listings and fancvrb make things nice and easy:

```
\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:
   }{}
 }
```

17.3 \oldstylenums

\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 }
 }
```

(*End definition for* `\oldstylenums` *and* `\liningnums`*. These functions are documented on page* ??.)

⟨/fontspec⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

\# 1995, 1996, 2027
\, ... 1791, 2720, 2721, 2722, 2723, 2724
\- 2, 106, 1738, 4140
@@ commands:
\@@_DeclareFontShape:nnnnnn 1590, 1597, 1608, 1626
\g_@@_OT_features_prop 81, 1783, 1785
\@@_add_nfssfont:nnnn . 1413, 1414, 1415, 1416, 1417, 1418, 1433, 2408
\@@_aff_error:n 2151, 2462, 2503, 2535
\l_@@_alias_bool 38, 819, 826, 832, 837, 844, 864
\l_@@_all_features_clist 65, 341, 1120, 1196, 1206, 1220, 1379
\g_@@_all_keyval_modules_clist 59, 821, 839, 2141
\g_@@_all_opentype_feature_names_prop 82, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088
\l_@@_arg_tl 2390, 2391, 2393, 2398, 2403
\l_@@_atsui_bool 24, 910, 1314, 1452, 1459, 1698
\l_@@_basename_tl . 31, 107, 1109, 1430
\g_@@_bf_series_seq 57, 2243, 2263, 2266
\g_@@_bfmathrm_tl 122, 675, 676, 3919, 3924, 3925, 3926
\@@_calc_scale:n . . . 2415, 2416, 2421
\g_@@_cfg_bool 33, 386, 387, 3970
\l_@@_check_bool 22, 444, 445, 1936, 1941, 1947, 1963
\l_@@_check_feat_bool 1829, 1831, 1832
\@@_check_lang:Nn 1885
\@@_check_lang:NnTF 999, 1014, 1885, 3021, 3043, 3050
\@@_check_ot_feat:Nn 1921
\@@_check_ot_feat:NnTF 950, 969, 1690, 1834, 1921
\@@_check_script:Nn 1853
\@@_check_script:NnTF 982, 1273, 1853, 2987, 3004
\@@_combo_sc_shape:n 1598, 1602, 1647, 1655
\@@_construct_font_call:nn 1069, 1232, 1234, 1237, 1239, 1242, 1382, 1494, 1495, 1515, 1584
\@@_construct_font_call:nnnnnn 1242, 1251
\l_@@_curr_bfname_tl 2261, 2269, 2271, 2274, 2321
\l_@@_curr_fontname_tl 108, 1424, 1425
\g_@@_curr_series_tl . . . 113, 1733, 2242, 2265, 2268, 2271, 2274, 2321
\@@_declare_shape:nnnn . . . 1505, 1518
\@@_declare_shape_loginfo:nn 1530, 1630
\@@_declare_shape_slanted:nn 1529, 1618
\@@_declare_shapes_normal:nn 1527, 1588
\@@_declare_shapes_smcaps:nn 1528, 1593
\g_@@_default_fontopts_clist 58, 732, 1208
\@@_define_aat_feature:nnnn 798, 2091, 3611, 3612, 3613, 3614, 3615, 3616, 3617, 3618, 3619, 3620, 3621, 3622, 3623, 3624, 3625, 3626, 3627, 3628, 3629, 3630, 3639, 3640, 3641, 3642, 3643, 3645,

3646, 3647, 3648, 3649, 3650, 3651, 3652, 3654, 3655, 3656, 3657, 3658, 3659, 3660, 3661, 3662, 3663, 3665, 3666, 3667, 3669, 3670, 3671, 3672, 3674, 3675, 3676, 3711, 3712, 3713, 3714, 3715, 3716, 3718, 3719, 3720, 3721, 3722, 3723, 3724, 3726, 3727, 3728, 3729, 3730, 3731, 3732, 3733, 3735, 3736, 3737, 3738, 3739, 3740, 3741, 3742, 3743, 3744, 3745, 3746
\@@_define_aat_feature_group:n 793, 2089, 3610, 3638, 3644, 3653, 3664, 3668, 3673, 3677, 3689, 3700, 3710, 3717, 3725, 3734
\@@_define_opentype_feature:nnnnn 808, 1779, 1815, 1816, 1817, 1821, 1822, 2726, 2747, 2760, 2779, 2795, 2811, 2823, 2829, 2830, 2831, 2833, 2838, 2839, 2840, 2843, 2866, 2867, 2869, 2889
\@@_define_opentype_feature_-group:n 803, 1775, 2725, 2746, 2759, 2778, 2794, 2810, 2822, 2832, 2842, 2868, 2888, 2906, 2915, 2928, 2941, 2962, 2971
\@@_define_opentype_onoffreset:nnnnn 1811, 2732, 2733, 2734, 2735, 2736, 2737, 2752, 2753, 2754, 2755, 2756, 2757, 2758, 2769, 2770, 2771, 2772, 2773, 2777, 2788, 2789, 2790, 2791, 2792, 2793, 2804, 2805, 2806, 2807, 2808, 2809, 2818, 2819, 2820, 2821, 2828, 2841, 2856, 2857, 2858, 2859, 2860, 2861, 2862, 2863, 2864, 2865, 2880, 2881, 2882, 2883, 2884, 2885, 2886, 2887, 2899, 2900, 2901, 2902, 2903, 2904, 2905, 2907, 2908, 2909, 2910, 2911, 2912, 2913, 2914
\@@_define_opentype_onreset:nnnnn 1819, 2745
\l_@@_defining_encoding_bool 41, 3757, 3763, 3769, 3775, 3781, 3805, 3807, 3812, 3818, 3824, 3830
\l_@@_disable_defaults_bool 37, 765, 1194
\l_@@_em_int 48, 4068, 4071, 4072, 4075, 4077, 4080, 4131, 4134
\g_@@_em_normalise_slant_bool 43, 4037, 4043, 4063
\g_@@_em_prop 83, 4035, 4052, 4053, 4069, 4077
\l_@@_em_switch_tl 4077, 4078
\l_@@_em_tmp_tl 4069, 4071
\l_@@_emdef_int 49, 4036, 4052, 4053, 4054
\l_@@_emshape_query_tl 4061, 4065, 4069, 4072
\@@_error:n 151, 1540, 3848
\@@_error:nn 152, 153, 1235, 1516, 2154, 2531, 3758, 3764, 3770, 3776, 3782, 3813, 3819, 3825, 3831
\g_@@_euenc_bool 35, 388, 389, 402, 415, 418, 437
\l_@@_ext_filename_tl 105, 1182, 1183, 1186, 1187
\l_@@_extension_tl 104, 1150, 1169, 1253, 2179, 2185, 2230
\l_@@_extensions_clist 62, 869, 870, 1145, 1158
\l_@@_external_bool 39, 1478, 2168, 2180
\@@_extract_all_features: 1191
\@@_extract_all_features:n 1119, 1191
\l_@@_fake_embolden_tl 127, 2679, 2682, 2696
\l_@@_fake_slant_tl 126, 2674, 2701, 2704
\l_@@_family_fontopts_clist 64, 1202, 1203, 1209
\l_@@_family_label_tl 125, 1052, 1059, 1202, 1204
\@@_feat_off:n 1811, 1816
\@@_feat_prop_add:nn 1779, 1791, 2720, 2721, 2722, 2723, 2724
\@@_feat_reset:n . . . 1812, 1817, 1822
\@@_find_autofonts: 1370, 1390
\l_@@_firsttime_bool 18, 887, 1128, 1269, 1706, 1726, 2341, 2459, 2549, 2560, 2572, 2626, 2672, 2694
\@@_font_is_file: . . 1068, 1259, 2170
\@@_font_is_name: 1259, 1727
\l_@@_font_path_tl 106, 1265, 1728, 2169
\@@_font_suppress_not_found_-error: 480, 495, 883, 1104
\l_@@_fontcfg_bool 28, 29, 1178, 2158, 2162
\l_@@_fontfeat_bf_clist 72, 1414, 2318, 2697
\l_@@_fontfeat_bfit_clist 74, 1417, 2329, 2681, 2683, 2703, 2705
\l_@@_fontfeat_bfsl_clist 76, 1418, 2337
\l_@@_fontfeat_clist 69, 888, 1227, 1301

\l_@@_fontfeat_curr_clist 70, 1549, 1558, 1571
\l_@@_fontfeat_it_clist 73, 1415, 2325, 2675
\l_@@_fontfeat_sc_clist 77, 1549, 2343
\l_@@_fontfeat_sl_clist 75, 1416, 2333
\l_@@_fontfeat_up_clist 71, 1413, 2314, 2349
\l_@@_fontid_tl 31, 103, 1120, 1122, 1336, 1338, 1340, 1345, 1348, 1358, 1363, 2384
\l_@@_fontname_bf_tl 129, 1172, 1395, 1401, 1414, 2269, 2698
\l_@@_fontname_bfit_tl 131, 1173, 1394, 1395, 1396, 1417, 2291, 2684, 2706
\l_@@_fontname_bfsl_tl 133, 1409, 1418, 2299
\l_@@_fontname_it_tl 130, 1171, 1394, 1406, 1415, 2286, 2676
\l_@@_fontname_sc_tl 134, 1553, 1565, 2309
\l_@@_fontname_sl_tl 132, 1409, 1416, 2295
\l_@@_fontname_tl . . 115, 764, 766, 770
\l_@@_fontname_up_tl 31, 35, 128, 1108, 1223, 1232, 1234, 1235, 1237, 1239, 2247, 2251
\@@_fontname_wrap:n 36, 1244, 1245, 1261, 1265
\l_@@_fontopts_clist 63, 1199, 1200, 1210, 1505, 1513, 1514
\g_@@_fontopts_prop 78, 744, 747, 751, 752, 1184, 1199, 1202, 1513
\@@_get_features:Nn 48, 889, 1127, 1297, 1578
\l_@@_graphite_bool 27, 1314, 1455, 1468
\g_@@_hexcol_tl . . 140, 142, 1327, 1749
\l_@@_hexcol_tl 102, 1326, 1328, 1749, 2540, 2544, 2557
\@@_if_autofont:nn 1492
\@@_if_autofont:nnTF 1484
\@@_if_detect_external:n 1155
\@@_if_detect_external:nTF 1068, 1111, 1155
\@@_if_font_feature:n 879
\@@_if_font_feature:nTF 877
\@@_if_merge_shape:n 4000
\@@_if_merge_shape:nTF . . . 1082, 3996
\@@_info:n 157, 1284, 2438, 3965
\@@_info:nn 158, 1486
\@@_info:nnn 159, 1373
\@@_init: 884, 1067, 1105, 1722
\@@_init_fontface: 1300, 1744
\@@_init_ttc:n 1116, 1167
\@@_int_mult_truncate:Nn . . 453, 2569
\@@_iv_str_to_num:Nn 967, 968, 1013, 1754, 1859, 1891, 1934
\@@_iv_str_to_num:w . 1761, 1762, 1764
\@@_keys_define_code:nnn . . 2147, 2153, 2156, 2160, 2164, 2176, 2177, 2186, 2218, 2223, 2228, 2235, 2240, 2245, 2249, 2253, 2278, 2289, 2293, 2297, 2301, 2312, 2316, 2323, 2327, 2331, 2335, 2339, 2346, 2351, 2357, 2361, 2365, 2369, 2373, 2377, 2381, 2388, 2411, 2457, 2483, 2504, 2508, 2512, 2536, 2566, 2582, 2586, 2592, 2603, 2607, 2611, 2712, 2716, 3033
\l_@@_keys_leftover_clist . . . 67, 1221, 1224, 1225, 1226, 1302, 1303, 1307, 1309, 1313, 1315, 1319, 1320
\@@_keys_set_known:nnN . 446, 1219, 1224, 1226, 1301, 1303, 1439, 1503
\l_@@_lang_name_tl . 137, 290, 1276, 1278, 1281, 1288, 1290, 1293, 2226
\l_@@_language_int 45, 945, 968, 1386, 1932, 1939, 3024, 3036, 3045, 3052
\l_@@_leftover_clist . . 66, 1503, 1505
\@@_load_external_fontoptions:Nn 1117, 1176, 1512
\@@_load_font: 1125, 1229
\@@_load_fontname:n 1504, 1509, 1544, 1565
\@@_main_DeclareFontsExtensions:n 603, 867
\@@_main_IfFontFeatureActiveTF:nnn 607, 873
\@@_main_addfontfeatures:n 563, 567, 756
\@@_main_aliasfontfeature:nn 587, 816
\@@_main_aliasfontfeatureoption:nnn 591, 835
\@@_main_defaultfontfeatures:nnn 559, 722
\@@_main_fontspec:nn 515, 609
\@@_main_newAATfeature:nnnn . 575, 790
\@@_main_newfontface:nnn . . . 555, 718
\@@_main_newfontfamily:nnn . . 551, 705
\@@_main_newfontfeature:nn . . 571, 780

\@@_main_newopentypefeature:nnn 579, 583, 800
\@@_main_setboldmathrm:nn . . 535, 673
\@@_main_setmainfont:nn 519, 547, 616
\@@_main_setmathrm:nn 531, 667
\@@_main_setmathsf:nn 539, 679
\@@_main_setmathtt:nn 543, 685
\@@_main_setmonofont:nn 527, 650
\@@_main_setsansfont:nn 523, 633
\@@_make_AAT_feature:nn 2095, 2098, 3685, 3696
\@@_make_AAT_feature_string:Nnn 2110
\@@_make_AAT_feature_string:NnnTF 912, 1699, 2103, 2110
\@@_make_OT_feature:nnn 1806, 1824, 1849, 2920, 2924, 2936, 2947, 2968, 2977
\@@_make_font_shapes:Nnnnn 1425, 1500
\@@_make_ot_smallcaps:TF 1687
\@@_make_smallcaps:TF 1555, 1687, 1693
\l_@@_mapping_tl 139, 1323, 1324, 2584, 2588, 2741, 2742, 3635
\g_@@_math_bool 34, 384, 385, 3935, 3937, 3938, 3939, 3940, 3941, 3942, 3943, 3944, 3947, 3949, 3950, 3953, 3955, 3956, 3957, 3958, 3959, 3960, 3961, 3962, 3963
\g_@@_math_euler_bool . 30, 3847, 3872
\g_@@_math_lucida_bool 31, 3851, 3852, 3853, 3878
\g_@@_mathrm_tl 121, 669, 670, 702, 3911, 3912, 3914, 3915, 3918, 3921
\g_@@_mathsf_tl 123, 681, 682, 703, 3916, 3928
\g_@@_mathtt_tl 124, 687, 688, 704, 3917, 3929
\l_@@_mm_bool 26, 1454, 1462, 2619, 2624
\@@_msg_new:nnn . 163, 168, 173, 177, 198, 238, 244, 248, 258, 262, 267, 271, 276, 281, 286, 292, 296, 303, 307, 311, 315, 320, 324, 329, 334, 338, 346, 350, 354, 358, 362, 367, 372
\@@_msg_new:nnnn 165, 182, 191, 202, 212, 220, 228
\l_@@_never_check_bool 40, 886, 1855, 1887, 1923
\l_@@_nfss_enc_tl 112, 612, 622, 628, 639, 645, 656, 662, 713, 1371, 1590, 1597, 1626, 1734, 2379
\l_@@_nfss_fam_tl . . 1343, 1345, 2383
\l_@@_nfss_prop 79, 2273, 2320
\l_@@_nfss_sc_tl 98, 1522, 1570, 1595, 1598, 1657
\l_@@_nfss_tl 97, 1521, 1545, 1591, 1645
\l_@@_nfssfont_prop . . . 80, 1420, 1443
\l_@@_nobf_bool 19, 1392, 1399, 2166, 2257, 2260, 2699
\l_@@_noit_bool 20, 1392, 1404, 2167, 2282, 2285, 2677
\l_@@_nosc_bool 21, 1551, 1562, 1568, 2305, 2308
\g_@@_opacity_tl 141, 143, 1327, 1748, 2558, 2570
\l_@@_opacity_tl 101, 1326, 1328, 1748, 2558, 2563, 2570, 2575
\l_@@_optical_size_tl 110, 1256, 1729, 2616, 2633
\l_@@_options_tl . . . 114, 763, 766, 770
\l_@@_ot_bool 25, 885, 928, 939, 965, 980, 993, 1010, 1025, 1040, 1305, 1453, 1465, 1473, 1667, 1695, 1725, 2614, 2624
\@@_ot_compat:nn 3063, 3067, 3068, 3069, 3070, 3071, 3072, 3073, 3074, 3075, 3076, 3077, 3078, 3079, 3080, 3081, 3082, 3083
\g_@@_pkg_euler_loaded_bool 32, 3837, 3840, 3846
\g_@@_postadjust_tl . . . 147, 148, 1750
\l_@@_postadjust_tl 146, 1591, 1598, 1627, 1658, 1660, 1750, 2506, 2516, 2528
\l_@@_pre_feat_sclist 117, 1383, 1585, 1664
\@@_preparse_features: . . . 1124, 1215
\l_@@_prev_unicode_name_tl 3803, 3808
\l_@@_primitive_font 496, 497
\@@_primitive_font_glyph_if_exist:Nn 501
\@@_primitive_font_glyph_if_exist:NnTF 501, 2526, 4154
\@@_primitive_font_gset:Nnn 472, 1238
\@@_primitive_font_if_exist:nTF 492, 1069
\@@_primitive_font_if_null:NTF 484, 497, 1235, 1516
\@@_primitive_font_if_null_p:N . 484
\@@_primitive_font_set:Nnn 472, 496, 1233, 1494, 1495, 1515
\@@_primitive_font_set_hyphenchar:Nn 509, 2517, 2529
\l_@@_proceed_bool . . 1828, 1837, 1841

\l_@@_punctspace_adjust_tl 144, 148, 1752, 2489, 2494, 2499
\l_@@_rawfeatures_sclist 47, 116, 889, 892, 1127, 1330, 1383, 1578, 1579, 1585, 1710, 1719, 1746
\@@_remove_clashing_featstr:n 1713, 1797, 1843
\g_@@_rmfamily_family 118, 618, 619, 623
\@@_sanitise_fontname:Nn 740, 1107, 1108, 1109, 1141, 1171, 1172, 1173, 1181
\@@_save_family:nn 1132, 1367
\@@_save_family_needed:n 1332
\@@_save_family_needed:nTF 1130, 1332
\@@_save_fontinfo:n 1369, 1375
\l_@@_saved_fontname_tl 109, 1523, 1536
\l_@@_scale_tl 100, 352, 1583, 1747, 2418, 2419, 2432
\l_@@_script_exist_bool 42, 2984, 2991, 2997
\l_@@_script_int 44, 942, 967, 996, 1013, 1385, 1893, 1898, 1931, 1939, 2990, 3008
\l_@@_script_name_tl 135, 290, 300, 1271, 1275, 1280, 1292, 2221
\@@_select_font_family:nn 766, 769, 1053, 1060, 1100, 1140
\@@_set_autofont:Nnn 1394, 1395, 1396, 1401, 1406, 1409, 1476
\@@_set_default_features:nn . 725, 729
\@@_set_faces: 1372, 1411
\@@_set_faces_aux:nnnnn . . 1420, 1422
\@@_set_font_default_features:nnn 726, 734
\@@_set_font_dimen:NnN 2429, 2430, 2441
\@@_set_font_type:N 909, 927, 938, 964, 979, 992, 1009, 1024, 1039, 1236, 1447
\@@_set_scriptlang: 1126, 1267
\@@_setboldmathrm_hook:nn . . 677, 697
\@@_setmainfont_hook:nn 629, 691
\@@_setmathrm_hook:nn 671, 694
\@@_setmathsf_hook:nn 683, 695
\@@_setmathtt_hook:nn 689, 696
\@@_setmonofont_hook:nn 663, 693
\@@_setsansfont_hook:nn 646, 692
\@@_setup_nfss:Nnnn . 1545, 1570, 1574
\@@_setup_single_size:nn . 1525, 1533
\g_@@_sffamily_family 119, 635, 636, 640
\@@_shape_merge:nn 1084, 1604, 1605, 3985, 3986, 3987, 3988, 3989, 3990, 3991, 3992, 3993, 3997, 4003, 4008
\g_@@_single_feat_tl 93, 881, 893, 895, 897, 1708, 2198, 2213, 2992, 3025, 3037, 3047, 3054
\l_@@_size_tl 95, 1535, 1540, 1541, 1576, 1583, 2371
\l_@@_sizedfont_tl 96, 1536, 1544, 1546, 2375
\l_@@_sizefeat_clist 60, 61, 1438, 1444, 2348, 2353
\l_@@_sizing_leftover_clist 68, 1539, 1545, 1571
\l_@@_smcp_shape_tl 1084, 1087, 1090, 1093
\@@_strip_leading_sign:Nw 1756, 1758
\@@_strip_plus_minus:n 809, 811
\@@_strip_plus_minus_aux:Nq . 811, 812
\l_@@_strnum_int 46, 1859, 1865, 1891, 1898, 1934, 1940, 2990, 3008, 3024, 3045, 3052
\l_@@_strong_int . . 50, 4109, 4112, 4113, 4116, 4118, 4121, 4132, 4135
\g_@@_strong_prop 84, 4091, 4099, 4100, 4110, 4118
\l_@@_strong_switch_tl . . . 4118, 4119
\l_@@_strong_tmp_tl 4110, 4112
\l_@@_strongdef_int 51, 4092, 4099, 4100, 4101
\@@_swap_plus_minus:n 1844, 1850
\@@_swap_plus_minus_aux:Nq 1850, 1851
\l_@@_tfm_bool 23, 1451, 1457
\l_@@_this_feat_tl . . 2391, 2404, 2409
\l_@@_this_font_tl 99, 1435, 1441, 1444, 2354, 2355, 2359, 2392, 2403, 2409
\l_@@_tmp_int 47, 2568, 2569, 2577, 2578
\l_@@_tmp_tl 94, 739, 740, 744, 747, 751, 752, 762, 941, 942, 944, 945, 1027, 1028, 1042, 1043, 1351, 1352, 1354, 1355, 1356, 1360, 1439, 1878, 1879, 1880, 1911, 1912, 1915, 1954, 1955, 1959
\l_@@_tmpa_bool . . 36, 1160, 1163, 1165
\l_@@_tmpa_dim 54, 2429, 2434
\l_@@_tmpa_fp 52
\l_@@_tmpb_dim 55, 2430, 2435
\l_@@_tmpb_fp 53
\l_@@_tmpb_tl 744, 745, 746, 747
\l_@@_tmpc_dim 56

\l_@@_tmpf_tl 995, 996
\@@_trace:n 160
\l_@@_ttc_index_tl 111, 1254, 1730, 2232, 2233, 2237, 2238
\g_@@_ttfamily_family 120, 652, 653, 657
\@@_update_featstr:n 786, 1324, 1328, 1703, 1846, 2105, 2510, 2605, 2609, 2621, 2640, 2648, 2657, 2714, 2718
\@@_warning:n 154, 383, 408, 414, 2101, 2231, 2596, 2600, 2627, 2665
\@@_warning:nn 155, 776, 833, 865, 1836, 2107, 2201, 2206, 2520, 2550, 2561, 2573, 3001, 3006, 3011, 3028, 3057
\@@_warning:nnn . . . 156, 796, 806, 1133
\l_@@_wordspace_adjust_tl 145, 148, 1751, 2467, 2475
\\ . 11, 179, 187, 188, 251, 252, 253, 264, 300, 317, 326, 331, 341, 342, 343, 364, 369, 375, 376, 377, 1634, 1646, 1660

A

\acute 3856
\addfontfeature 22, 37, 250, 565, 4234, 4238
\addfontfeatures 230, 561, 756
\advance 4211
\aliasfontfeature 585, 816, 2175, 2565, 2927, 2940, 3699
\aliasfontfeatureoption 589, 835, 2774, 2775, 2776, 3065
\AtBeginDocument . 97, 107, 432, 3969, 4181

B

\bar 3860
\bfdefault 39, 61, 1414, 1417, 1418, 1638, 1641, 1642, 1650, 1652, 1654, 2265, 2266, 2268, 3915, 3918, 3921, 3925, 3928, 3929
\bfseries 4136
\bgroup 4175
\boldmath 19
bool commands:
\bool_if:NTF 402, 418, 437, 832, 864, 910, 928, 939, 965, 980, 993, 1010, 1025, 1040, 1165, 1178, 1194, 1269, 1305, 1399, 1404, 1478, 1551, 1568, 1667, 1695, 1698, 1706, 1832, 1841, 1855, 1872, 1881, 1887, 1905, 1917, 1923, 1947, 1963, 2180, 2341, 2459, 2549, 2560, 2572, 2614, 2619, 2626, 2672, 2694, 2997, 3757, 3763, 3769, 3775, 3781, 3812, 3818, 3824, 3830, 3846, 3872, 3878, 3963, 3970, 4063
\bool_if:nTF 1314, 1392, 1620, 1760, 2624
\bool_lazy_and:nnTF 4002
\bool_new:N 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43
\bool_set_false:N . . 385, 387, 389, 445, 819, 837, 887, 1128, 1160, 1451, 1452, 1453, 1454, 1455, 1725, 1831, 1837, 1862, 1895, 1936, 2158, 2162, 2260, 2285, 2308, 2677, 2699, 2984, 3807, 3840, 3935, 3937, 3938, 3939, 3940, 3941, 3942, 3943, 3944, 3947, 3949, 3950, 3953, 3955, 3956, 3957, 3958, 3959, 3960, 3961, 3962, 4043
\bool_set_true:N 29, 384, 386, 388, 415, 444, 765, 826, 844, 885, 886, 1163, 1457, 1459, 1462, 1465, 1468, 1473, 1562, 1726, 1828, 1829, 1866, 1899, 1941, 2166, 2167, 2168, 2257, 2282, 2305, 2991, 3805, 3837, 3847, 3851, 3852, 3853, 4037
\bool_until_do:nn . . 1863, 1896, 1937
\breve 3861

C

\char 3751, 4144, 4155
char commands:
\char_set_catcode_active:n 4166, 4168
\char_set_catcode_ignore:n 380
\char_set_catcode_space:n 167
\check 3862
clist commands:
\clist_clear:N . . 888, 1200, 1203, 1514
\clist_count:N 2393
\clist_count:n 254
\clist_map_break: . . 1151, 1163, 2993
\clist_map_inline:Nn 821, 839, 1145, 1158
\clist_map_inline:nn 736, 1525, 1716, 2923, 2934, 2954, 2985, 3683, 3695, 4049, 4096
\clist_new:N 58, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70
\clist_put_left:Nn 1558
\clist_put_right:Nn 731, 2349, 2675, 2681, 2683, 2697, 2703, 2705
\clist_set:Nn . . . 61, 731, 869, 1196, 1206, 1438, 2141, 2314, 2318, 2325, 2329, 2333, 2337, 2343, 2348, 2353
\clist_set_eq:NN 1549

\colon 98, 98, 3868, 3869
\convertcolorspec 2540
cs commands:
\cs:w 739
\cs_end: 739
\cs_generate_variant:Nn . 161, 162, 452, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 1056, 1532, 1617, 1774, 1849
\cs_gset:Npn 4167
\cs_if_eq:NNTF 1877, 1910, 1953
\cs_if_exist:NTF 406, 903, 1090, 1354, 1364, 2538
\cs_if_exist_p:N 4005
\cs_new:Nn 163, 165, 446, 453, 509, 729, 734, 811, 1050, 1057, 1141, 1167, 1176, 1191, 1215, 1229, 1259, 1263, 1267, 1367, 1375, 1390, 1411, 1422, 1433, 1447, 1476, 1500, 1509, 1518, 1533, 1574, 1588, 1593, 1602, 1608, 1618, 1630, 1703, 1713, 1744, 1775, 1779, 1789, 1811, 1812, 1813, 1819, 1824, 1850, 2089, 2091, 2098, 2147, 2151, 2421, 2441, 2981, 3018, 3063, 3931, 3985, 3994, 4134, 4135, 4152, 4158
\cs_new:Npn 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 444, 445, 812, 1851, 2944
\cs_new_protected:Nn 1100
\cs_new_protected:Npn 4033, 4089
\cs_set:Nn 609, 616, 633, 650, 667, 673, 679, 685, 705, 718, 722, 756, 780, 790, 800, 816, 835, 867, 873, 1242, 1249, 1297, 1427, 1687, 1688, 1693, 1754, 3842
\cs_set:Npn 440, 472, 476, 480, 1265, 1722, 1758, 1764, 2463, 2555, 4086, 4087, 4088, 4127, 4128, 4188, 4192, 4198, 4204, 4207, 4209, 4218, 4225
\cs_set_eq:NN 439, 442, 691, 692, 693, 694, 695, 696, 697, 1073, 1140, 1261, 4169, 4222, 4229
\cs_to_str:N 707, 712, 739, 1879, 1912, 1955
\cs_undefine:N . 1613, 2384, 2386, 3832
\cyrillicencoding 91, 430, 434

D

\ddot 3858
\DeclareDocumentCommand 3755, 3761, 3767, 3773, 3779, 3785, 3810, 3816, 3822, 3828
\DeclareErrorFont 3788
\DeclareFontEncoding 409, 3787
\DeclareFontFamily 1371, 3790
\DeclareFontsExtensions 601, 867
\DeclareFontShape 42, 43, 1615, 3792, 3794, 3796, 3798, 3800
\DeclareFontSubstitution 410, 3789
\DeclareMathAccent 3856, 3857, 3858, 3859, 3860, 3861, 3862, 3863, 3864, 3865
\DeclareMathDelimiter 3903, 3904, 3905, 3906, 3907
\DeclareMathSymbol 3869, 3874, 3875, 3876, 3877, 3880, 3881, 3882, 3883, 3884, 3885, 3886, 3887, 3888, 3889, 3890, 3891, 3892, 3893, 3894, 3895, 3896, 3897, 3898, 3899, 3900, 3901, 3902, 3908
\DeclareOption 382, 384, 385, 386, 387, 388, 389, 390, 395
\DeclareRobustCommand 620, 637, 654, 710, 3979, 4013, 4018, 4023, 4028, 4058, 4105, 4140
\DeclareSymbolFont 3854, 3911
\DeclareSymbolFontAlphabet 3913
\DeclareTextCommand 3753, 3759, 3765
\DeclareTextComposite 3777
\DeclareTextCompositeCommand 3783
\DeclareTextFontCommand 3984, 4085, 4126
\DeclareTextSymbol 3771
\DeclareUnicodeAccent 3752
\DeclareUnicodeEncoding 171, 3785
\def 4151, 4172, 4180
\defaultfontfeatures 557, 722
\Delta 3891
dim commands:
\dim_compare:nNnTF 2444
\dim_new:N 54, 55, 56
\dim_set:Nn 2443
\dim_to_fp:n 2434, 2435
dim internal commands:
__dim_eval:w 455
__dim_eval_end: 455
\directlua 5, 1880, 1913, 1956
\discretionary 4142
\do 4176
\dospecials 4176
\dot 3864

E

\else 488, 1770, 1771, 1868, 1901, 1943, 4146, 4174
else commands:
\else: 505
\em 4058, 4085
\emfontdeclare 4033, 4137
\eminnershape 4085, 4137
\emph 4085
\emreset 4049, 4081, 4085
\emshape 4085, 4137
\EncodingAccent 3761
\EncodingCommand 3755
\EncodingComposite 3773
\EncodingCompositeCommand 3779
\encodingdefault . . 65, 628, 645, 662, 2425
\EncodingSymbol 3767
\endinput 6, 8, 15
environments:
listingcont* 4204
verbatim* 4188
\ERORROR 1310
etex commands:
\etex_iffontchar:D 503
\ExecuteOptions 400
exp commands:
\exp_after:wN 1538
\exp_args:Nnnx 808
\exp_args:Nnx 1815, 1816, 1817, 1821, 1822
\exp_args:No 1186
\exp_args:NV 897
\exp_args:Nx 1161, 1525, 1843
\exp_args:Nxx 1076
\exp_not:N 170, 622, 623, 624, 639, 640, 641, 656, 657, 658, 710, 712, 713, 714, 892, 893, 1180, 1299, 1635, 1647, 1660, 2944, 2945, 2948, 2956, 2957
\exp_not:n 620, 637, 654, 876, 1157, 1299, 1634, 1646, 1826

F

\familydefault 627, 644, 661
\FancyVerbSpace 4222
\fi 490, 1463, 1470, 1770, 1771, 1870, 1903, 1945, 3870, 3935, 3947, 3953, 4148, 4174
fi commands:
\fi: 507
file commands:
\file_if_exist:nTF 404, 1186
\file_input:n 1187
\font 31, 474, 478, 909, 912, 927, 938, 950, 964, 969, 979, 982, 992, 999, 1009, 1014, 1024, 1039, 1061, 1077, 1877, 1910, 1953, 2429, 2448, 2469, 2470, 2471, 2477, 2478, 2479, 2490, 2495, 2500, 2517, 2529, 4144, 4147, 4154
font commands:
\l_fontspec_font 31, 31, 1061, 1233, 1235, 1236, 1238, 1240, 1273, 1515, 1516, 1690, 1699, 1834, 2103, 2430, 2526, 2987, 3004, 3021, 3043, 3050
\l_tmpa_font 1494, 1496
\l_tmpb_font 1495, 1496
\fontdimen 66, 67, 2443, 2469, 2470, 2471, 2477, 2478, 2479, 2490, 2495, 2500
\fontdimen8 66
\fontencoding . 612, 622, 639, 656, 713, 2425
\fontfamily 20, 623, 640, 657, 712, 773, 2426
\fontname 1061, 1077, 1496
\fontshape 3982, 3997, 3998
\fontspec 20, 21, 22, 31, 107, 513, 609
fontspec commands:
\fontspec_calc_scale:n 65
\fontspec_complete_fontname:Nn 1424, 1427, 2247, 2251, 2261, 2286, 2291, 2295, 2299, 2309, 2375
\g_fontspec_default_fontopts_tl 21, 22
\l_fontspec_defined_shapes_tl 89, 344, 1632, 1732
\g_fontspec_encoding_tl 86, 420, 421, 423, 427, 428, 430, 431, 434, 435, 1734, 3911, 3912, 3914, 3915, 3916, 3917, 3918, 3921, 3924, 3925, 3926, 3928, 3929, 4161
\l_fontspec_family_tl 20, 31, 85, 340, 773, 1054, 1062, 1363, 1364, 1371, 1377, 1378, 1379, 1380, 1385, 1386, 1387, 1388, 1590, 1597, 1626, 1627, 2385, 2386
\l_fontspec_feature_string_tl 2105, 2136
\fontspec_font_if_exist:n 1064
\fontspec_font_if_exist:nTF . . . 1073
\l_fontspec_fontname_tl . . 31, 35, 88, 200, 264, 269, 274, 279, 284, 289, 294, 299, 352, 360, 1107, 1117, 1120, 1199, 1223, 1382, 1396, 1401, 1406,

1413, 1512, 1513, 1515, 1520, 1523, 1576, 1584, 2676, 2684, 2698, 2706
\l_fontspec_hyphenchar_tl 2523, 2524, 2526, 2529
\fontspec_if_aat_feature:nn 905
\fontspec_if_aat_feature:nnTF .. 905
\fontspec_if_current_feature:n 1074
\fontspec_if_current_feature:nTF 897, 1074
\fontspec_if_current_language:n 1035
\fontspec_if_current_language:nTF 1035
\fontspec_if_current_script:n . 1020
\fontspec_if_current_script:nTF 1020
\fontspec_if_feature:n 934
\fontspec_if_feature:nnn 960
\fontspec_if_feature:nnnTF 960
\fontspec_if_feature:nTF 934
\fontspec_if_fontspec_font: 901
\fontspec_if_fontspec_font:TF .. 759, 901, 907, 925, 936, 962, 977, 990, 1007, 1022, 1037
\fontspec_if_language:n 988
\fontspec_if_language:nn 1005
\fontspec_if_language:nnTF 1005
\fontspec_if_language:nTF 988
\fontspec_if_opentype: 923
\fontspec_if_opentype:TF 923
\fontspec_if_script:n 975
\fontspec_if_script:nTF 975
\fontspec_if_small_caps: 1080
\fontspec_if_small_caps:TF 1080
\l_fontspec_lang_tl 138, 948, 1388, 1672, 1683, 1960, 3023, 3035, 3046, 3053
\fontspec_maybe_setup_maths: .. 3931
\fontspec_merge_shape:n 3994, 4016, 4021, 4026, 4031
\l_fontspec_mode_tl 1679, 1737, 2209, 2213
\fontspec_new_lang:nn 599, 3018
\fontspec_new_script:nn ... 595, 2981
\fontspec_parse_colour:niii 2547, 2555
\fontspec_parse_cv:w 2944, 2956
_fontspec_parse_wordspace:w ... 2460, 2463
\fontspec_patch_fancyvrb: 4185, 4218
\fontspec_patch_listings: 4186, 4225
\fontspec_patch_moreverb: 4184, 4204
\fontspec_patch_verbatim: 4183, 4188
\fontspec_print_visible_spaces: 4165, 4180, 4194, 4200, 4214
\l_fontspec_renderer_tl 41, 87, 1255, 1460, 1466, 1469, 1731, 2194, 2198
\l_fontspec_script_tl 136, 947, 997, 1012, 1387, 1669, 1671, 1680, 1682, 1915, 1960, 2989, 3007
\fontspec_select:nn 20, 20, 1140
\fontspec_set_em_level:n 4134
\fontspec_set_family:Nnn 611, 618, 635, 652, 669, 670, 675, 676, 681, 682, 687, 688, 707, 1050
\fontspec_set_fontface:NNnn ... 1057
\fontspec_set_strong_level:n .. 4135
\fontspec_setup_maths: ... 3835, 3966
\fontspec_tmp: 439, 442
\fontspec_visible_space: 4152, 4169, 4222, 4229
\fontspec_visible_space_fallback: 4156, 4158
fontspec internal commands:
\l__fontspec_check_bool 1862, 1866, 1872, 1881, 1895, 1899, 1905, 1917
__fontspec_update_featstr:n .. 3706
\FontspecSetCheckBoolFalse 444
\FontspecSetCheckBoolTrue 444
fp commands:
\fp_eval:n 2434
\fp_new:N 52, 53

G

\g 99, 100
\Gamma 3890
\global 478, 4211
\grave 3857
group commands:
\group_begin: 494, 761, 882, 1066, 1103, 1502, 1611, 2423, 3866, 4047, 4094, 4165, 4194
\group_end: 498, 499, 772, 890, 1070, 1071, 1138, 1506, 1614, 2439, 3871, 4056, 4103, 4171

H

\hat 98, 3863
\hbox 4174
\hyphenchar 4144, 4147

I

\IfBooleanTF 731, 742

\ifcase 1456
\IfFontExistsTF 1073
\IfFontFeatureActiveTF 605, 873
\ifmmode 4174
\IfNoValueTF 724
\ifnum 1461, 1865, 1898, 1939, 4144
\ifx . 486, 1770, 1771, 3868, 3935, 3947, 3953
\ignorespaces . . 614, 631, 648, 665, 727, 778
\ImportEncoding 175
\InputIfFileExists 3972
int commands:
\int_case:nn 2196, 2211
\int_case:nnTF 2449
\int_compare:nTF 254, 2116, 2192, 2393, 2543, 2546, 2577
\int_compare_p:nNn . . 1863, 1896, 1937
\int_gincr:N 1355
\int_if_even:nTF 2121
\int_incr:N 1869, 1902, 1944, 4054, 4075, 4101, 4116
\int_new:N 44, 45, 46, 47, 48, 49, 50, 51, 1356
\int_set:Nn . 455, 457, 942, 945, 996, 1738, 1766, 1860, 1867, 1892, 1900, 1928, 1942, 2568, 2990, 3008, 3024, 3045, 3052, 4071, 4112, 4134, 4135
\int_set_eq:NN 482
\int_to_hex:n 2578
\int_use:N 1360, 4053, 4068, 4072, 4077, 4100, 4109, 4113, 4118
\int_zero:N 1739, 1740, 1741, 1861, 1894, 1935, 3036, 4036, 4080, 4092, 4121, 4131, 4132
\c_one 482, 4211
\l_tmpa_int 1861, 1863, 1865, 1867, 1869, 1894, 1896, 1898, 1900, 1902, 1935, 1937, 1940, 1942, 1944
\l_tmpb_int 1860, 1863, 1867, 1892, 1896, 1900, 1928, 1937, 1942
\c_zero 1461
\itdefault 1415, 1417, 1622, 1623, 1627, 1639, 1641, 3914, 3921, 3926, 3977, 3986, 3988, 3990, 4016
\itscdefault 1651, 1652, 3977, 3982, 3986, 3988, 3990, 3991, 3992
\itshape 4013, 4041, 4087

K

keys commands:
\l_keys_choice_int . . 2192, 2196, 2211
\keys_define:nn 782, 827, 845, 852, 859, 1777, 1794, 1801, 2090, 2093, 2149, 2187, 2636, 2644, 2653, 2663, 2668, 2690, 2739, 2916, 2929, 2950, 2963, 2972, 2980, 2983, 3017, 3020, 3039, 3631, 3678, 3690, 3701
\keys_if_choice_exist:nnnTF . 795, 805
\keys_if_exist:nnTF 792, 802, 823, 841, 849, 856
\l_keys_key_tl 273, 278, 283, 288
\keys_set:nn 3, 458, 828, 846, 853, 860, 1112, 1280, 1281, 1292, 1293, 1309, 1315, 1320, 2172, 2182, 2220, 2225, 2598, 3029, 3058
\keys_set_known:nn 1114
\keys_set_known:nnN 449, 459, 762, 1538
\l_keys_value_tl . . . 273, 278, 283, 288

L

\l . . . 22, 41, 43, 43, 51, 52, 52, 53, 53, 53, 57
\Lambda 3893
\latinencoding 92, 431, 435
\leavevmode 4174
\let 4176
\liningnums 4232
listingcont* (environment) 4204
luatex commands:
\luatex_postexhyphenchar:D 1741
\luatex_posthyphenchar:D 1739
\luatex_preexhyphenchar:D 1740
\luatex_prehyphenchar:D 1738

M

\mathalpha 3856, 3857, 3858, 3859, 3860, 3861, 3862, 3863, 3864, 3865, 3880, 3881, 3882, 3883, 3884, 3885, 3886, 3887, 3888, 3889, 3890, 3891, 3892, 3893, 3894, 3895, 3896, 3897, 3898, 3899, 3900
\mathbf 100, 210, 3915, 3925
\mathbin 3901
\mathchardef 3867
\mathclose 3874, 3877, 3904, 3906
\mathdollar 3908
\mathit 210, 3914, 3921, 3926, 4015
\mathopen 3903, 3905
\mathord 3907, 3908
\mathpunct 3869, 3876
\mathrel 3875, 3902
\mathring 3865
\mathrm 19, 99, 3913, 3924
\mathsf 3916, 3928
\mathtt 3917, 3929

\mddefault 1413, 1415, 1416, 1637, 1639, 1640, 1649, 1651, 1653, 3911, 3912, 3914, 3916, 3917, 3924, 3926
msg commands:
\msg_error:nn 151
\msg_error:nnn 152, 153
\msg_fatal:nn 14
\msg_info:nn 157
\msg_info:nnn 158
\msg_info:nnnn 159
\msg_line_context: 250
\msg_new:nnn 9, 161, 164
\msg_new:nnnn 162, 166
\msg_redirect_module:nnn 392, 393, 397, 398
\msg_redirect_name:nnn 2597
\msg_trace:nn 160
\msg_warning:nn 154
\msg_warning:nnn 155
\msg_warning:nnnn 156

N

\newAATfeature 573, 790
\NewDocumentCommand 513, 517, 521, 525, 529, 533, 537, 541, 545, 549, 553, 557, 561, 565, 569, 573, 577, 581, 585, 589, 593, 597, 601, 605, 4236
\newfontface 21, 553, 718
\newfontfamily 549, 705, 720
\newfontfeature 22, 23, 23, 569, 780
\newfontlanguage 597, 3228, 3229, 3230, 3231, 3232, 3233, 3234, 3235, 3236, 3237, 3238, 3239, 3240, 3241, 3242, 3243, 3244, 3245, 3246, 3247, 3248, 3249, 3250, 3251, 3252, 3253, 3254, 3255, 3256, 3257, 3258, 3259, 3260, 3261, 3262, 3263, 3264, 3265, 3266, 3267, 3268, 3269, 3270, 3271, 3272, 3273, 3274, 3275, 3276, 3277, 3278, 3279, 3280, 3281, 3282, 3283, 3284, 3285, 3286, 3287, 3288, 3289, 3290, 3291, 3292, 3293, 3294, 3295, 3296, 3297, 3298, 3299, 3300, 3301, 3302, 3303, 3304, 3305, 3306, 3307, 3308, 3309, 3310, 3311, 3312, 3313, 3314, 3315, 3316, 3317, 3318, 3319, 3320, 3321, 3322, 3323, 3324, 3325, 3326, 3327, 3328, 3329, 3330, 3331, 3332, 3333, 3334, 3335, 3336, 3337, 3338, 3339, 3340, 3341, 3342, 3343, 3344, 3345, 3346, 3347, 3348, 3349, 3350, 3351, 3352, 3353, 3354, 3355, 3356, 3357, 3358, 3359, 3360, 3361, 3362, 3363, 3364, 3365, 3366, 3367, 3368, 3369, 3370, 3371, 3372, 3373, 3374, 3375, 3376, 3377, 3378, 3379, 3380, 3381, 3382, 3383, 3384, 3385, 3386, 3387, 3388, 3389, 3390, 3391, 3392, 3393, 3394, 3395, 3396, 3397, 3398, 3399, 3400, 3401, 3402, 3403, 3404, 3405, 3406, 3407, 3408, 3409, 3410, 3411, 3412, 3413, 3414, 3415, 3416, 3417, 3418, 3419, 3420, 3421, 3422, 3423, 3424, 3425, 3426, 3427, 3428, 3429, 3430, 3431, 3432, 3433, 3434, 3435, 3436, 3437, 3438, 3439, 3440, 3441, 3442, 3443, 3444, 3445, 3446, 3447, 3448, 3449, 3450, 3451, 3452, 3453, 3454, 3455, 3456, 3457, 3458, 3459, 3460, 3461, 3462, 3463, 3464, 3465, 3466, 3467, 3468, 3469, 3470, 3471, 3472, 3473, 3474, 3475, 3476, 3477, 3478, 3479, 3480, 3481, 3482, 3483, 3484, 3485, 3486, 3487, 3488, 3489, 3490, 3491, 3492, 3493, 3494, 3495, 3496, 3497, 3498, 3499, 3500, 3501, 3502, 3503, 3504, 3505, 3506, 3507, 3508, 3509, 3510, 3511, 3512, 3513, 3514, 3515, 3516, 3517, 3518, 3519, 3520, 3521, 3522, 3523, 3524, 3525, 3526, 3527, 3528, 3529, 3530, 3531, 3532, 3533, 3534, 3535, 3536, 3537, 3538, 3539, 3540, 3541, 3542, 3543, 3544, 3545, 3546, 3547, 3548, 3549, 3550, 3551, 3552, 3553, 3554, 3555, 3556, 3557, 3558, 3559, 3560, 3561, 3562, 3563, 3564, 3565, 3566, 3567, 3568, 3569, 3570, 3571, 3572, 3573, 3574, 3575, 3576, 3577, 3578, 3579, 3580, 3581, 3582, 3583, 3584, 3585, 3586, 3587, 3588, 3589, 3590, 3591, 3592, 3593, 3594, 3595, 3596, 3597, 3598, 3599, 3600, 3601, 3602, 3603, 3604, 3605, 3606, 3607, 3608, 3609
\newfontscript 593, 3084, 3085, 3086, 3087, 3088, 3089, 3090, 3091, 3092, 3093, 3094, 3095, 3096, 3097, 3098, 3099, 3100, 3101, 3102, 3103, 3104, 3105, 3106, 3107, 3108, 3109, 3110, 3111, 3112, 3113, 3114, 3115, 3116, 3117, 3118, 3119,

3120, 3121, 3122, 3123, 3124, 3125, 3126, 3127, 3128, 3129, 3130, 3131, 3132, 3133, 3134, 3135, 3136, 3137, 3138, 3139, 3140, 3141, 3142, 3143, 3144, 3145, 3146, 3147, 3148, 3149, 3150, 3151, 3152, 3153, 3154, 3155, 3156, 3157, 3158, 3159, 3160, 3161, 3162, 3163, 3164, 3165, 3166, 3167, 3168, 3169, 3170, 3171, 3172, 3173, 3174, 3175, 3176, 3177, 3178, 3179, 3180, 3181, 3182, 3183, 3184, 3185, 3186, 3187, 3188, 3189, 3190, 3191, 3192, 3193, 3194, 3195, 3196, 3197, 3198, 3199, 3200, 3201, 3202, 3203, 3204, 3205, 3206, 3207, 3208, 3209, 3210, 3211, 3212, 3213, 3214, 3215, 3216, 3217, 3218, 3219, 3220, 3221, 3222, 3223, 3224, 3225, 3226, 3227
\newICUfeature 581
\newopentypefeature 577, 800
\normalfont . 630, 647, 664, 4048, 4095, 4130
\normalsize 429, 1612
\null 4174
\nullfont 486
\numexpr 2128

O

\oldstylenums 2, 108, 4232
\Omega 3900
\or 1458, 1464, 1467

P

\par 4212
Path 2164
\Phi 3898
\Pi 3895
prg commands:
\prg_new_conditional:Nnn 501, 879, 901, 905, 923, 934, 960, 975, 988, 1005, 1020, 1035, 1064, 1074, 1080, 1155, 1332, 1492, 1853, 1885, 1921, 2110, 4000
\prg_return_false: . . 489, 498, 506, 895, 898, 903, 913, 916, 920, 928, 931, 950, 953, 957, 969, 971, 973, 982, 984, 986, 999, 1001, 1003, 1014, 1016, 1018, 1029, 1031, 1033, 1044, 1046, 1048, 1071, 1078, 1095, 1098, 1165, 1365, 1497, 1872, 1881, 1905, 1917, 1947, 1963, 2114, 2134, 4011
\prg_return_true: 487, 499, 504, 898, 903, 913, 928, 950, 969, 982, 999, 1014, 1029, 1044, 1070, 1078, 1096, 1165, 1365, 1498, 1856, 1872, 1881, 1888, 1905, 1917, 1924, 1947, 1963, 2137, 4011
\prg_set_conditional:Nnn . . . 484, 492
\ProcessOptions 401
prop commands:
\prop_clear:N 4035, 4091
\prop_get:NnN 763, 764, 941, 944, 947, 948, 995, 997, 1027, 1042
\prop_get:NnNTF 464, 465, 744, 1199, 1202, 1513, 4069, 4077, 4110, 4118
\prop_gput:Nnn 463, 747, 1378, 1379, 1380, 1385, 1386, 1387, 1388, 1785, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 4053, 4100
\prop_gput_if_new:Nnn 462, 4052, 4099
\prop_gremove:Nn 751
\prop_if_in:NnTF 1184, 1783
\prop_map_inline:Nn 1420
\prop_new:N 78, 79, 80, 81, 82, 83, 84, 1377
\prop_put:Nnn 460, 461, 752, 1443, 2273, 2320
\providecommand 3747, 3748, 3749, 3750, 3751, 3752, 3977, 3978
\Psi 3899

Q

quark commands:
\q_nil . . 811, 812, 1756, 1758, 1761, 1762, 1764, 1850, 1851, 2945, 2957
\q_stop 2460, 2463

R

\relax 2128, 3751, 3867, 3981, 4020, 4025, 4030, 4174
\RenewDocumentCommand 4232
\RequirePackage 4, 427, 441
\RequirePackageWithOptions 6, 8
\rmdefault . 20, 99, 424, 619, 627, 702, 2426
\rmfamily 65, 65, 620, 2448

S

scan commands:
\scan_stop: 474, 478, 503, 511, 1061, 4155
\scdefault 1604, 1605, 1606, 1649, 1650, 3977, 3978, 3986, 3987, 3988, 3989, 3992, 3993, 4026
\scshape 4013
\select 12
\selectfont 613, 624, 641, 658, 714, 773, 2427, 3982, 3997, 3998
seq commands:
\seq_gput_right:Nn 2243
\seq_if_empty:NTF 2263
\seq_new:N 57
\seq_put_right:Nn 2266
\setboldmathrm 19, 100, 533, 673, 699
\setmainfont 16, 19, 97, 517, 616
\SetMathAlphabet 3914, 3915, 3916, 3917, 3921, 3924, 3925, 3926, 3928, 3929
\setmathrm 99, 529, 667, 698
\setmathsf 537, 679, 700
\setmathtt 541, 685, 701
\setmonofont 525, 650
\setromanfont 545
\setsansfont 521, 633
\SetSymbolFont 3855, 3912, 3918
\settoheight 2446
\sfdefault 425, 636, 644, 703
\sffamily 637
\Sigma 3896
\sishape 3977
\sldefault 1416, 1418, 1623, 1626, 1640, 1642, 3978, 3987, 3989, 3991, 4021
\slscdefault 1653, 1654, 3978, 3987, 3989, 3990, 3991, 3993
\slshape 4013, 4039
\space 352, 2118, 2123, 2128
str commands:
\c_backslash_str 3833
\c_colon_str 2945, 2957
\str_case:nn 1635, 1647, 1852
\str_case:nnTF 814, 2413
\str_case_x:nnTF 2485
\str_if_eq:nnTF 466, 1028, 1043, 2448, 2514, 2594, 2999
\str_if_eq_p:nn 1760
\str_if_eq_x:nnTF 627, 644, 661, 1169, 1247, 1326, 1496, 2230
\str_if_eq_x_p:nn 1622, 1623
\str_lower_case:n 1169, 2230
\string 171, 175, 210, 230, 250
\strong 4126
\strongenv 4105, 4126
\strongfontdeclare 4089, 4136
\strongreset 4096, 4122, 4126
sys commands:
\sys_if_engine_luatex:TF 3
\sys_if_engine_xetex:TF 7

T

TEX and LATEX 2ε commands:
\@ 22, 49, 49, 51, 52
\@@sverb 4178, 4180
\@filelist 429
\@ifpackageloaded 3835, 3844, 3851, 3852, 3853, 3933, 3937, 3938, 3939, 3940, 3941, 3942, 3943, 3944, 3945, 3949, 3950, 3951, 3955, 3956, 3957, 3958, 3959, 3960, 3961, 3962, 4190, 4206, 4220, 4227
\@ifstar 4178
\@makeother 4176
\@noligs 4177
\@nomath 4060, 4107
\@onlypreamble 698, 699, 700, 701
\@sverb 4180
\@sxverbatim 4200
\@tempa 3867, 3868
\@verb 4178
\@verbatim 4194, 4200, 4214
\add@unicode@accent . 3751, 3753, 3765
\color@ 2538
\curr@fontshape 1878, 1911, 1954
\define@antt@mathversions 3935
\define@iwona@mathversions 3947
\define@kurier@mathversions . . . 3953
\f@encoding 1090, 1093, 1094, 4007
\f@family 611, 763, 764, 903, 941, 944, 947, 948, 995, 997, 1027, 1042, 1090, 1093, 1094, 4007
\f@series 1090, 1093, 1094, 4007, 4099, 4110, 4113, 4161

\f@shape 1084, 3997, 4003, 4008, 4052, 4061, 4161
\f@size 1234, 1239, 1494, 1495, 1515, 1613, 1878, 1911, 1954
\listing@line 4211
\lst@visiblespace 4229
\not@math@alphabet 3981, 4015, 4020, 4025, 4030
\reset@font 4128
\thelisting@line 4211
\two@digits 2936, 2948
\verb@eol@error 4176
\verbatim@font 4177
\verbatim@line 4212
\verbatim@processline 4209
\verbatim@start 4194, 4214
\xlx@defaulthyphenchar ... 4145, 4151
\z@ 4144
tex commands:
\tex_hyphenchar:D 511
\textsi 3977
\textvisiblespace 4162
\the 4212
\Theta 3892
\tilde 3859
tl commands:
\c_empty_tl 1761, 1762, 1770, 1771
\tl_clear:N 745, 1204, 1521, 1522, 1535, 1728, 1729, 1730, 1731, 1732, 1733, 1746, 1747, 1751, 1752, 2185, 2392, 2404, 2742
\tl_const:Nn 3986, 3987, 3988, 3989, 3990, 3991, 3992, 3993
\tl_count:n 2543, 2546
\tl_gclear:N 881
\tl_gput_right:Nn 1632, 1710
\tl_gremove_all:Nn 1719
\tl_gset:Nn 1358, 1363, 1708, 2198, 2213, 2242, 2265, 2379, 2432, 2992, 3025, 3037, 3047, 3054
\tl_gset_eq:NN 1734
\tl_if_empty:NTF 895, 1271, 1276, 1288, 1323, 1348, 1441, 1460, 1466, 1469, 1482, 1540, 1553, 1595, 1669, 1680, 2113, 2130, 2133, 2354, 2679, 2701, 3919
\tl_if_empty:nTF ... 467, 468, 469, 750, 1437, 1480, 1658, 1781, 1792, 1831, 2100, 2255, 2280, 2303, 2465
\tl_if_eq:NNTF 1092, 2558, 2570
\tl_if_eq:nnTF 470, 2268
\tl_if_exist:NTF 1338, 1343, 1604, 2385
\tl_if_exist_p:N 4003
\tl_if_in:NnTF 429, 1147, 2398
\tl_if_in:nnTF . 1076, 1162, 4039, 4041
\tl_if_single:nTF 738, 2522
\tl_new:N . 71, 72, 73, 74, 75, 76, 77, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 144, 145, 146, 147, 1340
\tl_put_left:Nn 2130
\tl_put_right:Nn 746, 1581, 2506, 2516, 2528
\tl_remove_all:Nn 870, 1144, 1183, 1352, 1431
\tl_remove_once:Nn 1149
\tl_replace_all:Nnn .. 471, 1430, 4065
\tl_set:Nn 142, 143, 148, 420, 421, 423, 424, 425, 426, 702, 703, 704, 739, 1012, 1052, 1059, 1087, 1120, 1143, 1150, 1182, 1275, 1278, 1290, 1351, 1429, 1435, 1460, 1466, 1469, 1485, 1664, 1737, 1878, 1879, 1911, 1912, 1954, 1955, 2112, 2118, 2123, 2126, 2136, 2169, 2179, 2194, 2209, 2221, 2226, 2232, 2233, 2237, 2238, 2355, 2359, 2371, 2383, 2390, 2418, 2419, 2467, 2475, 2489, 2494, 2499, 2523, 2524, 2544, 2557, 2563, 2575, 2584, 2588, 2616, 2633, 2674, 2696, 2741, 2989, 3007, 3023, 3035, 3046, 3053, 3635, 3804, 4061
\tl_set_eq:NN 428, 430, 431, 434, 435, 619, 628, 636, 645, 653, 662, 1054, 1062, 1084, 1223, 1330, 1345, 1523, 1536, 1748, 1749, 1750, 2269, 2391, 2403, 2676, 2684, 2698, 2706, 3803, 3808
\tl_to_str:N 1120
\tl_to_str:n 1077, 3833
\tl_trim_spaces:n 164, 166
\tl_use:N 1605, 3997, 4008
\l_tmpa_tl 2112, 2113, 2136
\l_tmpb_tl 2118, 2123, 2126, 2130, 2133, 2136
token commands:
\token_to_str:N 2538, 3833

\tracingall 1308
\ttdefault 426, 653, 661, 704
\ttfamily 654
\typeout 448, 450, 758, 766, 818, 825, 843, 851, 858, 875, 876, 892, 893, 1102, 1122, 1136, 1157, 1180, 1193, 1217, 1231, 1232, 1237, 1299, 1307, 1313, 1319, 1335, 1336, 1449, 1511, 1520, 1541, 1546, 1557, 1561, 1576, 1579, 1610, 1705, 1709, 1715, 1718, 1724, 1805, 1826, 1927, 1952, 2271, 2396, 2401, 3974, 4068, 4072, 4109, 4113

U

\UndeclareAccent 3810
\UndeclareCommand 3810
\UndeclareComposite 3828
\UndeclareSymbol 3810
\UndeclareTextCommand . . 3814, 3820, 3826
\unexpanded 1193
\UnicodeEncodingName 3759, 3765, 3771, 3777, 3783, 3803, 3804, 3808, 3814, 3820, 3826, 3833
\UnicodeFontFile 3747, 3793, 3795, 3797, 3799, 3801
\UnicodeFontName 3748
\UnicodeFontTeXLigatures 3749, 3750, 3793, 3795, 3797, 3799, 3801
\updefault 1094, 1413, 1414, 1637, 1638, 3911, 3912, 3915, 3916, 3917, 3918, 3924, 3925, 3928, 3929, 3992, 3993, 4031
\upshape 4013, 4086, 4088
\Upsilon 3897
use commands:
\use:N 253, 712
\use:n 620, 637, 654, 708, 767, 1261, 1538, 2942
\use_i:nnn 2409
\use_ii:nnn 2409
\use_iii:nnn 2390
\use_none:nn 691, 692, 693, 694, 695, 696, 697
\usefont 4161
\UTFencname 90, 428

V

\verb 107, 4172
\verb* 106, 4172
verbatim* (environment) 4188

X

xetex commands:
\xetex_suppressfontnotfounderror:D 482
\XeTeXcountvariations 1461
\XeTeXfeaturename 2112
\XeTeXfonttype 1456
\XeTeXisexclusivefeature 2116
\XeTeXOTcountfeatures 1930
\XeTeXOTcountlanguages 1893
\XeTeXOTcountscripts 1860
\XeTeXOTfeaturetag 1939
\XeTeXOTlanguagetag 1898
\XeTeXOTscripttag 1865
\XeTeXpicfile 439, 440, 442
\XeTeXselectorname 2118, 2123, 2128
\Xi 3894