The fontspec package
Font selection for XƎLaTeX and LuaLaTeX

Will Robertson
With contributions by Khaled Hosny,
Philipp Gesang, Joseph Wright, and others.
`http://latex3.github.io/fontspec/`

2025/09/29 v2.9g

Contents

File I

fontspec.dtx

1 Package declaration

List all `dtx` files for running the `ins` file and typesetting the code.

```
⟨*dtx⟩
\gdef\FONTSPECDTX{
  \DTX{fontspec.dtx}
  \DTX{fontspec-code-load.dtx}
  \DTX{fontspec-code-vars.dtx}
  \DTX{fontspec-code-msg.dtx}
  \DTX{fontspec-code-opening.dtx}
  \DTX{fontspec-code-fontload.dtx}
  \DTX{fontspec-code-interfaces.dtx}
  \DTX{fontspec-code-user.dtx}
  \DTX{fontspec-code-api.dtx}
  \DTX{fontspec-code-internal.dtx}
  \DTX{fontspec-code-opentype.dtx}
  \DTX{fontspec-code-graphite.dtx}
  \DTX{fontspec-code-keyval.dtx}
  \DTX{fontspec-code-feat-opentype.dtx}
  \DTX{fontspec-code-scripts.dtx}
  \DTX{fontspec-code-lang.dtx}
  \DTX{fontspec-code-feat-aat.dtx}
  \DTX{fontspec-code-enc.dtx}
  \DTX{fontspec-code-math.dtx}
  \DTX{fontspec-code-closing.dtx}
  \DTX{fontspec-code-xfss.dtx}
}
⟨/dtx⟩
```

Now exit if we're using plain TEX; this would usually be the case when loading this file with `fontspec.ins`.

```
⟨*dtx⟩
\def\tmpa{plain}
\ifx\tmpa\fmtname\expandafter\endinput\fi
⟨/dtx⟩
```

Metadata for documentation; the official title and authors of the package.

```
⟨*dtx⟩
\title{
  The \textsf{fontspec} package\\
  Font selection for \XeLaTeX\ and \LuaLaTeX
}
\author{
   \textsc{Will Robertson}\\
   With contributions by Khaled Hosny,\\
   Philipp Gesang, Joseph Wright, and others.\\
   \url{http://latex3.github.io/fontspec/}
}
⟨/dtx⟩
```

Declare the package version and date for each of the `.sty` files generated. In addition, declare the version and date for this `.dtx` file.

```
⟨fontspec⟩\RequirePackage{xparse}
⟨fontspec & load⟩\ProvidesExplPackage{fontspec}%
⟨fontspec & XE⟩\ProvidesExplPackage{fontspec-xetex}%
⟨fontspec & LU⟩\ProvidesExplPackage{fontspec-luatex}%
⟨*dtx⟩
\ProvidesExplFile{fontspec.dtx}
⟨/dtx⟩
⟨*fontspec⟩
  {2025/09/29}{2.9g}{Font selection for XeLaTeX and LuaLaTeX}
⟨/fontspec⟩
```

Here the version and date are setup for typesetting the documentation.

```
⟨*dtx⟩
\GetFileInfo{fontspec.dtx}
\date{\filedate \qquad \fileversion}
⟨/dtx⟩
```

1.1 Lua header

```
⟨lua⟩fontspec          = fontspec or {}
⟨lua⟩local fontspec    = fontspec
⟨lua⟩fontspec.module   = {
⟨lua⟩    name          = "fontspec",
⟨lua⟩    version       = "2.9g",
⟨lua⟩    date          = "2025/09/29",
⟨lua⟩    description   = "Font selection for XeLaTeX and LuaLaTeX",
⟨lua⟩    author        = "Khaled Hosny, Philipp Gesang, Will Robertson",
⟨lua⟩    copyright     = "Khaled Hosny, Philipp Gesang, Will Robertson",
⟨lua⟩    license       = "LPPL v1.3c"
⟨lua⟩}
```

File II

fontspec-code-load.dtx

1 The `fontspec.sty` loading file

Before we begin, for the rest of the package we use the `@@` expl3 module syntax with module name '`fontspec`'.

```
⟨@@=fontspec⟩
```

The `fontspec.sty` file is simply set up to load the appropriate `fontspec-xetex.sty` or `fontspec-luatex.sty` file. This is performed by the following code.

```
⟨*load⟩
```

LuaLaTeX

```
\sys_if_engine_luatex:T
  {
    \RequirePackage{luaotfload}
    \lua_now:e{require("fontspec")}
    \RequirePackage{fontspec-luatex}
    \endinput
  }
```

XeLaTeX

```
\sys_if_engine_xetex:T
  {
    \RequirePackage{fontspec-xetex}
    \endinput
  }
```

Other If not one of the above, error and exit.

```
\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"~ instead~ of~ "latex"~ or~ "pdflatex".
  }
\msg_fatal:nn {fontspec} {cannot-use-pdftex}
```

Closing That's the end of the `fontspec.sty` file.

```
\endinput
⟨/load⟩
```

File III

fontspec-code-vars.dtx

1 Variables

`\l_@@_firsttime_bool`

As `\keys_set:nn` is run multiple times, some of its information storing only occurs once while we decide if the font family has been defined or not. When the later processing is occurring per-shape this no longer needs to happen; this is indicated by the 'firsttime' conditional.

`\l_@@_never_check_bool`

Used to disable checking opentype script, language, and tags when running checking code that has a user-defined return path we want to allow the higher-level code to dictate the logic. TODO: tidy this up!

2 Implementation

This file consists solely of declaration of variables used by fontspec. In some cases these variables are also initialised with default values.

Booleans for package options For package options:

```
⟨*options⟩
\bool_new:N \g_@@_cfg_bool
\bool_new:N \g_@@_math_bool
⟨/options⟩
⟨*fontspec⟩
```

Booleans for flow control

`\l_@@_firsttime_bool`

```
\bool_new:N \l_@@_firsttime_bool
```

(*End of definition for* `\l_@@_firsttime_bool`. *This function is documented on page 9.*)

`\l_@@_never_check_bool`

```
\bool_new:N \l_@@_never_check_bool
```

(*End of definition for* `\l_@@_never_check_bool`. *This function is documented on page 9.*)

Booleans for processing logic

```
\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_@@_harfbuzz_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

\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_@@_defining_encoding_bool
\bool_new:N \l_@@_scriptlang_exist_bool
\bool_new:N \g_@@_em_normalise_slant_bool
\bool_new:N \l_@@_external_kpse_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_@@_tmpa_int
\int_new:N \l_@@_tmpb_int
\int_new:N \l_@@_tmpc_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
```

Floats

```
\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
```

Sequences

```
\seq_new:N \l_@@_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
\clist_new:N \l_@@_arg_clist
\clist_new:N \l_@@_this_feat_clist

\clist_new:N \l_@@_fontfeat_up_clist
\clist_new:N \l_@@_fontfeat_bf_clist
\clist_new:N \l_@@_fontfeat_it_clist
\clist_new:N \l_@@_fontfeat_bfit_clist
\clist_new:N \l_@@_fontfeat_sl_clist
\clist_new:N \l_@@_fontfeat_bfsl_clist
\clist_new:N \l_@@_fontfeat_sw_clist
\clist_new:N \l_@@_fontfeat_bfsw_clist
\clist_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
\prop_new:N \g_@@_fontid_family_prop
\prop_new:N \g_@@_family_int_prop
\prop_new:N \g_@@_rawvariations_prop
```

Token lists

Visible (perhaps?)

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

2e interactions

```
\tl_clear_new:N \UTFencname
\tl_clear_new:N \cyrillicencoding
\tl_clear_new:N \latinencoding
```

Renderer/shaper

```
\tl_new:N \l_@@_renderer_tl
\tl_new:N \l_@@_mode_tl
\tl_new:N \l_@@_shaper_tl

\tl_new:N \g_@@_defined_shapes_tl
\tl_new:N \g_@@_single_feat_tl
\tl_new:N \l_@@_basename_tl
\tl_new:N \g_@@_curr_series_tl
\tl_new:N \l_@@_curr_fontname_tl
\tl_new:N \l_@@_curr_bfname_tl
\tl_new:N \l_@@_ext_filename_tl
\tl_new:N \l_@@_extension_tl
\tl_new:N \l_@@_font_path_tl
\tl_new:N \l_@@_fontid_tl
\tl_new:N \l_@@_fontname_tl
\tl_new:N \l_@@_options_tl
\tl_new:N \l_@@_saved_fontname_tl
\tl_new:N \l_@@_prev_unicode_name_tl

\tl_new:N \g_@@_nfss_enc_tl
\tl_new:N \g_@@_nfss_family_tl
\tl_new:N \l_@@_nfss_sc_tl
\tl_new:N \l_@@_nfss_tl
\tl_new:N \l_@@_nfss_fam_tl

\tl_new:N \l_@@_size_tl
\tl_new:N \l_@@_sizedfont_tl
\tl_new:N \l_@@_this_font_tl
\tl_new:N \l_@@_ttc_index_tl
\tl_new:N \l_@@_smcp_shape_tl
```

EM and STRONG

```
\tl_new:N \l_@@_emshape_query_tl
\tl_new:N \l_@@_em_switch_tl
\tl_new:N \l_@@_strong_switch_tl
```

Scratch variables

```
\tl_new:N \l_@@_tmp_tl
\tl_new:N \l_@@_tmpa_tl
\tl_new:N \l_@@_tmpb_tl
\tl_new:N \l_@@_em_tmp_tl
\tl_new:N \l_@@_strong_tmp_tl
```

Maths fonts

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

Defaults: (these are set elsewhere; TODO: check if redundant)

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

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

Internal font names

```
\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_sw_tl
\tl_new:N \l_@@_fontname_bfsw_tl
\tl_new:N \l_@@_fontname_sc_tl
```

Script and Language

```
\tl_new:N  \l_@@_script_tl
\tl_new:N  \l_@@_script_name_tl
\tl_set:Nn \l_@@_script_name_tl {CustomDefault}
```

```
\tl_new:N  \l_@@_lang_tl
\tl_new:N  \l_@@_lang_name_tl
\tl_set:Nn \l_@@_lang_name_tl {Default}
```

Generic font features

```
\tl_new:N \l_@@_scale_tl
\tl_new:N \l_@@_hyphenchar_tl
\tl_new:N \l_@@_hexcol_tl
\tl_new:N \l_@@_opacity_tl
\tl_new:N \l_@@_optical_size_tl
\tl_new:N \l_@@_mapping_tl
\tl_new:N \l_@@_punctspace_adjust_tl
\tl_new:N \l_@@_wordspace_adjust_tl
\tl_new:N \l_@@_postadjust_tl
\tl_new:N \g_@@_instance_tl
```

```
\tl_const:Nn \c_@@_hexcol_tl {000000}
⟨XE⟩ \tl_const:Nn \c_@@_opacity_tl {FF~}
⟨LU⟩ \tl_const:Nn \c_@@_opacity_tl {}
\tl_const:Nn \c_@@_postadjust_tl { \l_@@_wordspace_adjust_tl \l_@@_punctspace_adjust_tl }
```

Semi-colon-lists Not a real data structure but sensible to name accordingly.

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

Font families

```
\tl_new:N \l_@@_rmfamily_family_tl
\tl_new:N \l_@@_sffamily_family_tl
\tl_new:N \l_@@_ttfamily_family_tl
\tl_new:N \l_@@_rmfamily_encoding_tl
\tl_new:N \l_@@_sffamily_encoding_tl
\tl_new:N \l_@@_ttfamily_encoding_tl
```

File IV

fontspec-code-msg.dtx

1 Error/warning/info messages

```
\@@_msg_new:nn  \@@_msg_new:nn  {⟨msg id⟩} {⟨message⟩}
\@@_msg_new:nnn \@@_msg_new:nnn {⟨msg id⟩} {⟨message⟩} {⟨additional text⟩}
```

These functions define messages using the standard `\msg_new:nnn(n)` commands. Wrappers are used to pre-set the package name and to trim spaces, allowing form convenient definition later in this file.

An additional linebreak is added to avoid awkward breaks when the filename is included in the error message.

2 Implementation

2.1 Messages for package options

```
⟨*options⟩
\msg_new:nnn {fontspec} {cm-default-obsolete}
 {
  The~"cm-default"~package~option~is~obsolete.
 }
\msg_new:nnn {fontspec} {enc-obsolete}
 {
  The~"#1"~package~option~is~obsolete.~TU~is~the~default~encoding.
 }
\msg_new:nnn {fontspec} {math-opt-unknown}
 {
  The~ global~ option~ 'math=#1'~ is~ not~ recognised.~ It~ will~ be~ ignored.
 }
⟨/options⟩
```

2.2 Messages for general package behaviour

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 \@@_error:nxx   { \msg_error:nnxx   {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} }
```

\@@_msg_new:nn
\@@_msg_new:nnn

```
\cs_new:Nn \@@_msg_new:nn
  {
    \msg_new:nnx {fontspec} {#1} { ^^J \tl_trim_spaces:n {#2} }
  }
\cs_new:Nn \@@_msg_new:nnn
  {
    \msg_new:nnxx {fontspec} {#1} { ^^J \tl_trim_spaces:n {#2} }
                                  {     \tl_trim_spaces:n {#3} }
  }
```

(*End of definition for* `\@@_msg_new:nn` *and* `\@@_msg_new:nnn`*. These functions are documented on page 15.*)

Allow messages to be written with spaces acting as normal:

```
\char_set_catcode_space:n {32}
```

2.3 Errors

```
\@@_msg_new:nn {only-inside-encdef}
 {
  \exp_not:N #1 can only be used in the second argument
  to \string\DeclareUnicodeEncoding.
 }
\@@_msg_new:nn {no-size-info}
 {
  Size information must be supplied.\\
  For example, SizeFeatures={Size={8-12},...}.
 }
\@@_msg_new:nnn {font-not-found}
 {
  The font "#1" cannot be found; this may be but usually is not
  a fontspec bug. Either there is a typo in the font name/file,
  the font is not installed (correctly), or there is a bug
  in the underlying font loading engine (XeTeX/luaotfload).
 }
 {
  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:nnn {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:nn {no-glyph}
 {
  '#1' does not contain glyph #2.
 }
\@@_msg_new:nnn {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:nnn {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:nnn {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:nnn {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.
 }
```

2.4 Warnings

```
\@@_msg_new:nn {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:nn {tu-missing}
 {
  The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
 }
\@@_msg_new:nn {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:nn {feature-option-overwrite}
 {
  Option '#2' of font feature '#1' overwritten.
 }
\@@_msg_new:nn {ot-tag-too-long}
 {
  OpenType tag '#1' is too long; script, language, and feature tags must be four characters or
 }
\@@_msg_new:nn {aat-feature-not-exist}
 {
  '\l_keys_key_tl=\l_keys_value_tl' feature not supported
  for AAT font '\l_fontspec_fontname_tl'.
 }
\@@_msg_new:nn {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:nn {no-opticals}
 {
  '#1' doesn't appear to have an Optical Size axis.
 }
\@@_msg_new:nn {script-not-exist}
 {
  Script '#2' not explicitly supported within font '#1'.
  Check the typeset output, and if it is okay then ignore this warning.
  Otherwise a different font should be chosen.
 }
\@@_msg_new:nn {language-not-exist}
 {
  Language '#1' not explicitly supported
  within font '\l_fontspec_fontname_tl'
  with script '\l_@@_script_name_tl'.
  Check the typeset output, and if it is okay then ignore this warning.
  Otherwise a different font should be chosen.
 }
\@@_msg_new:nn {only-xetex-feature}
 {
  Ignored XeTeX-only feature: '#1'.
 }
\@@_msg_new:nn {only-luatex-feature}
 {
  Ignored LuaTeX-only feature: '#1'.
 }
\@@_msg_new:nn {unknown-renderer}
```

```
 {
  Renderer '#1' unknown. Assuming Harfbuzz with 'shaper=#1'.
  Please raise a fontspec issue to add this shaper to the interface.
 }
\@@_msg_new:nn {no-mapping}
 {
  Input mapping not supported in LuaTeX.
 }
\@@_msg_new:nn {no-mapping-ligtex}
 {
  Input mapping not supported in LuaTeX.\\
  Use "Ligatures=TeX" instead of "Mapping=tex-text".
 }
```

```
\@@_msg_new:nn {font-index-needs-ttc}
 {
  The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
  Feature ignored.
 }
\@@_msg_new:nn {feat-cannot-remove}
 {
  The "#1" feature cannot be deactivated. Request ignored.
 }
```

2.5 Info messages

```
\@@_msg_new:nn {defining-font}
 {
  Font family '\g_@@_nfss_family_tl' created for font '#2'
  with options [\l_@@_all_features_clist].\\
  \\
  This font family consists of the following NFSS series/shapes:\\
  \g_@@_defined_shapes_tl
 }
\@@_msg_new:nn {no-font-shape}
 {
  Could not resolve font "#1" (it probably doesn't exist).
 }
\@@_msg_new:nn {set-scale}
 {
  \l_fontspec_fontname_tl\space scale = \l_@@_scale_tl.
 }
\@@_msg_new:nn {setup-math}
 {
  Adjusting the maths setup (use [no-math] to avoid this).
 }
\@@_msg_new:nn {opa-twice}
 {
  Opacity set twice, in both Colour and Opacity.\\
  Using specification "Opacity=#1".
 }
\@@_msg_new:nn {opa-twice-col}
 {
  Opacity set twice, in both Opacity and Colour.\\
  Using an opacity specification in hex of "#1/FF".
 }
\@@_msg_new:nn {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}
```

File V

fontspec-code-opening.dtx

1 Opening code

`\FontspecSetCheckBoolTrue`
`\FontspecSetCheckBoolFalse`

Allows easy return of boolean results (`\l_@@_check_bool`) without requiring specific catcode regimes. Used within Lua functions.

`\@@_keys_set_known:nnN`
`\@@_keys_set_known:nxN`

Wrapper around standard `\keys_set_known:nnN` which permits debugging.

`\@@_int_mult_truncate:Nn`

`\@@_int_mult_truncate:nn` ⟨*int*⟩ {⟨*scale*⟩}

Scales ⟨*integer variable*⟩ by ⟨*scale*⟩, discarding any fractional components. E.g.,

```
% \int_new:N \foo
% \int_set:Nn \foo {3}
% \__fontspec_int_mult_truncate:Nn \foo {3.3}
% \int_show:N \foo % \foo = 9
%
% \int_set:Nn \foo {3}
% \__fontspec_int_mult_truncate:Nn \foo {3.4}
% \int_show:N \foo % \foo = 10
```

`\@@_lua_function:ne`
`\@@_lua_function:nee`
`\@@_lua_function:neee`
`\@@_lua_function:neeee`

`\@@_lua_function:ne` {⟨*method*⟩} {⟨*arg. 1*⟩}

Wrapper around `\lua_now:e`, hard-coded to refer to a method of the `fontspec` module. `\@@_lua_function:ne {foo} {bar}` executes `fontspec.foo("bar")` in the Lua engine. Variants for up to four arguments are provided.

2 Implementation

2.1 Package options

```
\DeclareKeys
  {
    ,math    .bool_gset:N = \g_@@_math_bool
    ,math    .usage:n     =  preamble
    ,math / unknown .code:n = { } % \msg_warning:nnn {fontspec} {math-opt-unknown} {#1}
    ,no-math .bool_gset_inverse:N = \g_@@_math_bool
    ,no-math .usage:n             =  preamble
    ,config  .bool_gset:N = \g_@@_cfg_bool
    ,config  .usage:n     = load
    ,no-config  .bool_gset_inverse:N = \g_@@_cfg_bool
    ,no-config  .usage:n             = load
```

```
    ,cm-default  .code:n = { \msg_warning:nn  {fontspec} {cm-default-obsolete} }
    ,euenc       .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{euenc} }
    ,tuenc       .code:n = { \msg_warning:nnn {fontspec} {enc-obsolete}{tuenc} }
```

```
    ,quiet .code:n =
      {
        \msg_redirect_module:nnn { fontspec } { warning } { info }
        \msg_redirect_module:nnn { fontspec } { info } { none }
      }
    ,silent .code:n =
      {
        \msg_redirect_module:nnn { fontspec } { warning } { none }
        \msg_redirect_module:nnn { fontspec } { info } { none }
      }
    ,verbose .code:n =
      {
        \msg_redirect_module:nnn { fontspec } { warning } { warning }
        \msg_redirect_module:nnn { fontspec } { info } { info }
      }
  }
```

```
\SetKeys{config,math}
\ProcessKeyOptions
```

2.2 Encodings

Now the default, with a just-in-case check:

```
\cs_if_exist:cF {T@TU}
  {
    \@@_warning:n {tu-clash}
    \DeclareFontEncoding{TU}{}{}
    \DeclareFontSubstitution{TU}{lmr}{m}{n}
  }
\tl_gset: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.

2.3 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 of definition for* `\FontspecSetCheckBoolTrue` *and* `\FontspecSetCheckBoolFalse`*. These functions are documented on page 21.*)

`\@@_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 of definition for* `\@@_keys_set_known:nnN`*. This function is documented on page 21.*)

`\@@_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 of definition for* `\@@_int_mult_truncate:Nn`*. This function is documented on page 21.*)

`\@@_lua_function:ne`
`\@@_lua_function:nee`
`\@@_lua_function:neee`
`\@@_lua_function:neeee`

```
⟨*LU⟩
\cs_set:Npn \@@_lua_function:ne    #1#2       { \lua_now:e { fontspec.#1 ("#2")
\cs_set:Npn \@@_lua_function:nee   #1#2#3     { \lua_now:e { fontspec.#1 ("#2","#3")
\cs_set:Npn \@@_lua_function:neee  #1#2#3#4   { \lua_now:e { fontspec.#1 ("#2","#3","#4")
\cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2","#3","#4","#5")
⟨/LU⟩
```

(*End of definition for* `\@@_lua_function:ne` *and others. These functions are documented on page 21.*)

2.4 expl3 variants

```
\cs_generate_variant:Nn \int_set:Nn {Nv}
\cs_generate_variant:Nn \prop_gput_if_not_in:Nnn {NeV}
\cs_generate_variant:Nn \prop_gput:Nnn  {Nxn} % needed by unicode-math
\cs_generate_variant:Nn \tl_if_empty:nF {f}
\cs_generate_variant:Nn \tl_if_eq:nnT {oe}

\cs_generate_variant:Nn \msg_new:nnn  {nnx}
\cs_generate_variant:Nn \msg_new:nnnn {nnxx}
```

File VI

fontspec-code-fontload.dtx

1 expl3 interface for primitive font loading

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

`\@@_primitive_font_set:Nnn ⟨font cs⟩ {⟨file/name⟩} {⟨size (dim. expr)⟩}`

Wrapper around TeX `\font` primitive.

`\@@_font_suppress_not_found_error:`

Wrapper around XeTeXLuaTeX primitive, used to be able to programmatically query font existence.

`\@@_primitive_font_if_null_p:N` ⋆
`\@@_primitive_font_if_null:NTF` ⋆

`\@@_primitive_font_if_null:NTF ⟨font cs⟩ {⟨true code⟩} {⟨false code⟩}`

Query whether the ⟨`font cs`⟩ is the `\nullfont`.

`\@@_primitive_font_set:NnnTF`
`\@@_primitive_font_gset:NnnTF`

`\@@_primitive_font_set:NnnTF ⟨font cs⟩ {⟨file/name⟩} {⟨size (dim. expr)⟩} {⟨true code⟩} {⟨false code⟩}`

Attempts to set the ⟨`font cs`⟩ and provide branching for success or failure according to whether the font exists.

`\@@_primitive_font_if_exist:nTF`

`\@@_primitive_font_if_exist:nTF {⟨file/name⟩} {⟨true code⟩} {⟨false code⟩}`

Attempts to load the font at the current font size minus 1 sp with a dummy metafont cs (`\l_@@_primitive_font`), and branches according to whether the font exists. This is intended to be used speculatively before potentially performing additional processing steps (e.g., an analysis of available font features) before setting up the 'true' font.

`\@@_primitive_font_glyph_if_exist_p:Nn` ⋆
`\@@_primitive_font_glyph_if_exist:NnTF` ⋆

`\@@_primitive_font_glyph_if_exist:nTF ⟨font cs⟩ {⟨char. number (int. expr)⟩} {⟨true code⟩} {⟨false code⟩}`

Wrapper around primitive `\iffontchar`.

`\@@_primitive_font_set_hyphenchar:Nn`

`\@@_primitive_font_set_hyphenchar:Nn ⟨font cs⟩ {⟨char. number (int. expr)⟩}`

Setter function around primitive `\hyphenchar`.

`\@@_primitive_font_get_name:N`
`\@@_primitive_font_current_name:`

`\@@_primitive_font_get_name:N ⟨font cs⟩`

Wrapper function around TeX primitive `\fontname`. The no-argument version is equivalent to `\fontname\font`.

2 Implementation

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

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

(*End of definition for* `\@@_primitive_font_set:Nnn` *and* `\@@_primitive_font_gset:Nnn`*. These functions are documented on page 24.*)

`\@@_font_suppress_not_found_error:`

```
\cs_set:Npn \@@_font_suppress_not_found_error:
  {
    \int_set:Nn \suppressfontnotfounderror {1}
  }
```

(*End of definition for* `\@@_font_suppress_not_found_error:`*. This function is documented on page 24.*)

`\@@_primitive_font_if_null_p:N`
`@@_primitive_font_if_null:NTF`

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

(*End of definition for* `\@@_primitive_font_if_null:NTF`*. This function is documented on page 24.*)

`\@@_primitive_font_set:NnnTF`
`\@@_primitive_font_gset:NnnTF`

```
\prg_new_conditional:Nnn \@@_primitive_font_set:Nnn {TF,T,F}
  {
    \@@_primitive_font_set:Nnn #1 {#2} {#3}
    \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
  }
\prg_new_conditional:Nnn \@@_primitive_font_gset:Nnn {TF,T,F}
  {
    \@@_primitive_font_gset:Nnn #1 {#2} {#3}
    \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
  }
\cs_set:Npn \@@_primitive_font_set:Onn   { \exp_last_unbraced:No \@@_primitive_font_set:Nnn }
\cs_set:Npn \@@_primitive_font_set:OnnF  { \exp_last_unbraced:No \@@_primitive_font_set:NnnF }
\cs_set:Npn \@@_primitive_font_gset:Onn  { \exp_last_unbraced:No \@@_primitive_font_gset:Nnn }
\cs_set:Npn \@@_primitive_font_gset:OnnF { \exp_last_unbraced:No \@@_primitive_font_gset:NnnF
```

(*End of definition for* `\@@_primitive_font_set:NnnTF` *and* `\@@_primitive_font_gset:NnnTF`*. These functions are documented on page 24.*)

\@@_primitive_font_if_exist:n*TF*

```
\prg_new_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} { \f@size pt - 1sp }
      \@@_primitive_font_if_null:NTF \l_@@_primitive_font
        { \group_end: \prg_return_false: }
        { \group_end: \prg_return_true:  }
  }
```

(*End of definition for* `\@@_primitive_font_if_exist:nTF`*. This function is documented on page 24.*)

\@@_primitive_font_glyph_if_exist_p:Nn
\@@_primitive_font_glyph_if_exist:Nn*TF*

```
\prg_new_conditional:Nnn \@@_primitive_font_glyph_if_exist:Nn {p,TF,T,F}
  {
    \tex_iffontchar:D #1 \int_eval:n {#2} \scan_stop:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End of definition for* `\@@_primitive_font_glyph_if_exist:NnTF`*. This function is documented on page 24.*)

\@@_primitive_font_set_hyphenchar:Nn

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

(*End of definition for* `\@@_primitive_font_set_hyphenchar:Nn`*. This function is documented on page 24.*)

\@@_primitive_font_get_name:N
\@@_primitive_font_current_name:

```
\cs_new_eq:NN \@@_primitive_font_get_name:N \fontname
\cs_new:Npn \@@_primitive_font_current_name:
  {
    \@@_primitive_font_get_name:N \tex_font:D
  }
```

(*End of definition for* `\@@_primitive_font_get_name:N` *and* `\@@_primitive_font_current_name:`*. These functions are documented on page 24.*)

File VII

fontspec-code-interfaces.dtx

1 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 1 on page 32.

`\fontspec`

`\fontspec [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Defines a dummy font with specified options and font name/file.

`\setmainfont`
`\setsansfont`
`\setmonofont`

`\setmainfont [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Defines the standard rm/sf/tt font families with specified options and font name/file.

`\setmathrm`
`\setboldmathrm`
`\setmathsf`
`\setmathtt`

`\setmathrm [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Defines the standard `\mathrm`/`sf`/`tt` font families with specified options and font name/file. `\setboldmathrm` sets the `\mathrm` font when `\boldmath` is active.

`\setromanfont`

`\setromanfont [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Deprecated alias for `\setmainfont`.

`\newfontfamily`
`\renewfontfamily`
`\setfontfamily`
`\providefontfamily`

`\setromanfont ⟨font switch cs⟩ [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Defines font family as specified, activated using the ⟨`font switch cs`⟩.

`\newfontface`
`\renewfontface`
`\setfontface`
`\providefontface`

`\setromanfont ⟨font switch cs⟩ [⟨options⟩] {⟨font name/file⟩} [⟨options⟩]`

Defines font family as specified, activated using the ⟨`font switch cs`⟩. Only the exact font specified will be defined, without attempting to find bold/italic/etc. shapes.

`\oldstylenums`
`\liningnums`

`{\oldstylenums 123} {\liningnums 123}` → 123 123

Re-implementations of font-switching commands which activate the necessaru font features (`onum` and `lnum` in OpenType).

2 Implementation

`\fontspec`

```
\NewDocumentCommand \fontspec { O{} m O{} }
  {
    \@@_main_fontspec:nn {#1,#3} {#2}
    \ignorespaces
  }
```

(*End of definition for* `\fontspec`. *This function is documented on page 27.*)

`\setmainfont`
`\setsansfont`
`\setmonofont`

```
\NewDocumentCommand \setmainfont { O{} m O{} }
  {
    \@@_main_setmainfont:nn {#1,#3} {#2}
    \ignorespaces
  }
\NewDocumentCommand \setsansfont { O{} m O{} }
  {
    \@@_main_setsansfont:nn {#1,#3} {#2}
    \ignorespaces
  }
\NewDocumentCommand \setmonofont { O{} m O{} }
  {
    \@@_main_setmonofont:nn {#1,#3} {#2}
    \ignorespaces
  }
```

(*End of definition for* `\setmainfont`, `\setsansfont`, *and* `\setmonofont`. *These functions are documented on page 27.*)

`\setmathrm`
`\setboldmathrm`
`\setmathsf`
`\setmathtt`

```
\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}
  }
```

(*End of definition for* `\setmathrm` *and others. These functions are documented on page 27.*)

`\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 of definition for* `\setromanfont`*. This function is documented on page 27.*)

`\newfontfamily`
`\renewfontfamily`
`\setfontfamily`
`\providefontfamily`

```
\NewDocumentCommand \newfontfamily { m O{} m O{} }
  {
    \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
  }
\NewDocumentCommand \renewfontfamily { m O{} m O{} }
  {
    \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
  }
\NewDocumentCommand \setfontfamily { m O{} m O{} }
  {
    \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
  }
\NewDocumentCommand \providefontfamily { m O{} m O{} }
  {
    \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
  }
```

(*End of definition for* `\newfontfamily` *and others. These functions are documented on page 27.*)

`\newfontface`
`\renewfontface`
`\setfontface`
`\providefontface`

```
\NewDocumentCommand \newfontface { m O{} m O{} }
  {
    \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
  }
\NewDocumentCommand \renewfontface { m O{} m O{} }
  {
    \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
  }
\NewDocumentCommand \setfontface { m O{} m O{} }
  {
    \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
  }
\NewDocumentCommand \providefontface { m O{} m O{} }
  {
    \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
  }
```

(*End of definition for* `\newfontface` *and others. These functions are documented on page 27.*)

`\defaultfontfeatures` This macro takes one argument that consists of all of feature options that will be applied by default to all subsequent `\fontspec` commands.

```
\NewDocumentCommand \defaultfontfeatures { t+ o m }
  {
    \IfNoValueTF {#2}
      { \@@_set_default_features:nn {#1} {#3} }
      { \@@_set_font_default_features:nnn {#1} {#2} {#3} }
    \ignorespaces
  }
```

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

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

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

`\oldstylenums` This is performed only after the preamble to overwrite any redefinition by textcomp:

```
\AtBeginDocument
  {
    \RenewDocumentCommand \oldstylenums {m}
      {
        \@@_main_oldstylenums:n {#1}
      }
  }
```

(*End of definition for* `\oldstylenums`. *This function is documented on page* 27.)

`\liningnums`

```
\NewDocumentCommand \liningnums {m}
  {
    \@@_main_liningnums:n {#1}
  }
```

(*End of definition for* `\liningnums`. *This function is documented on page* 27.)

File VIII

fontspec-code-user.dtx

1 User command internals

1.1 Font selection

`\@@_main_fontspec:nn` 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.

```
\cs_new:Nn \@@_main_fontspec:nn
  {
    \fontspec_set_family:Nnn \f@family {#1} {#2}
    \fontencoding { \g_@@_nfss_enc_tl }
    \selectfont
  }
```

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

`\rmfamily`
`\sffamily`
`\ttfamily`

Add an encoding switch to the three family commands.

```
\tl_set:Nn \l_@@_rmfamily_encoding_tl { \encodingdefault }
\tl_set:Nn \l_@@_sffamily_encoding_tl { \encodingdefault }
\tl_set:Nn \l_@@_ttfamily_encoding_tl { \encodingdefault }
```

```
\AddToHook{rmfamily}{ \fontencoding \l_@@_rmfamily_encoding_tl  }
\AddToHook{sffamily}{ \fontencoding \l_@@_sffamily_encoding_tl  }
\AddToHook{ttfamily}{ \fontencoding \l_@@_ttfamily_encoding_tl  }
```

(*End of definition for* `\rmfamily`, `\sffamily`, *and* `\ttfamily`. *These functions are 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).

They end with `\normalfont` so that if they're used in the document, the change registers immediately.

```
\cs_new:Nn \@@_main_setmainfont:nn
  {
⟨debug⟩\typeout{::~_main_setmainfont:nn}
    \DeclareFontSeriesDefault[rm]{bf}{\bfdefault}
    \fontspec_set_family:Nnn \l_@@_rmfamily_family_tl {#1} {#2}
    \tl_set_eq:NN \rmdefault \l_@@_rmfamily_family_tl
    \tl_set_eq:NN \l_@@_rmfamily_encoding_tl \g_@@_nfss_enc_tl
    \str_if_eq:eeT {\familydefault} {\rmdefault}
      { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
    \@@_setmainfont_hook:nn {#1} {#2} % for unicode-math only
    \normalfont
  }
```

(*End of definition for* `\setmainfont`. *This function is documented on page* 27.)

`\setsansfont` Same as above.

```
\cs_new:Nn \@@_main_setsansfont:nn
  {
    \DeclareFontSeriesDefault[sf]{bf}{\bfdefault}
    \fontspec_set_family:Nnn \l_@@_sffamily_family_tl {#1} {#2}
    \tl_set_eq:NN \sfdefault \l_@@_sffamily_family_tl
    \tl_set_eq:NN \l_@@_sffamily_encoding_tl \g_@@_nfss_enc_tl
    \str_if_eq:eeT {\familydefault} {\sfdefault}
      { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
    \@@_setsansfont_hook:nn {#1} {#2} % for unicode-math only
    \normalfont
  }
```

(*End of definition for* `\setsansfont`*. This function is documented on page 27.*)

`\setmonofont` Same as above.

```
\cs_new:Nn \@@_main_setmonofont:nn
  {
    \DeclareFontSeriesDefault[tt]{bf}{\bfdefault}
    \fontspec_set_family:Nnn \l_@@_ttfamily_family_tl {#1} {#2}
    \tl_set_eq:NN \ttdefault \l_@@_ttfamily_family_tl
    \tl_set_eq:NN \l_@@_ttfamily_encoding_tl \g_@@_nfss_enc_tl
    \str_if_eq:eeT {\familydefault} {\ttdefault}
      { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
    \@@_setmonofont_hook:nn {#1} {#2} % for unicode-math only
    \normalfont
  }
```

(*End of definition for* `\setmonofont`*. This function is documented on page 27.*)

`\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_new:Nn \@@_main_setmathrm:nn
  {
⟨XE⟩ \fontspec_gset_family:Nnn \g_@@_mathrm_tl {#1} {#2}
⟨LU⟩ \fontspec_gset_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
    \@@_setmathrm_hook:nn {#1} {#2} % for unicode-math only
  }
```

(*End of definition for* `\setmathrm`*. This function is documented on page 27.*)

`\setboldmathrm`

```
\cs_new:Nn \@@_main_setboldmathrm:nn
  {
⟨XE⟩ \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
⟨LU⟩ \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
    \@@_setboldmathrm_hook:nn {#1} {#2} % for unicode-math only
  }
```

(*End of definition for* `\setboldmathrm`*. This function is documented on page 27.*)

`\setmathsf`

```
\cs_new:Nn \@@_main_setmathsf:nn
  {
⟨XE⟩ \fontspec_gset_family:Nnn \g_@@_mathsf_tl {#1} {#2}
⟨LU⟩ \fontspec_gset_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
    \@@_setmathsf_hook:nn {#1} {#2} % for unicode-math only
  }
```

(*End of definition for* `\setmathsf`. *This function is documented on page* 27.)

`\setmathtt`

```
\cs_new:Nn \@@_main_setmathtt:nn
  {
⟨XE⟩ \fontspec_gset_family:Nnn \g_@@_mathtt_tl {#1} {#2}
⟨LU⟩ \fontspec_gset_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
    \@@_setmathtt_hook:nn {#1} {#2} % for unicode-math only
  }
```

(*End of definition for* `\setmathtt`. *This function is documented on page* 27.)

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

`\@@_main_newfontfamily:NnnN` The inner fontspec workings define a font family, which is then used in a typical NFSS `\fontfamily` declaration, saved in the macro name specified. The fourth argument determines which xparse function to set the macro with (new/renew/etc).

```
\cs_new:Nn \@@_main_newfontfamily:NnnN
  {
    \fontspec_set_family:cnn { l_@@_ \cs_to_str:N #1 _family_tl } {#2} {#3}
    \use:x
      {
        \exp_not:N #4 \exp_not:N #1 {}
          {
            \exp_not:N \fontfamily { \use:c { l_@@_ \cs_to_str:N #1 _family_tl } }
            \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
            \exp_not:N \selectfont
            }
        }
    }
```

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

`\@@_main_newfontface:NnnN`

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

```
\cs_new:Nn \@@_main_newfontface:NnnN
  {
    \@@_main_newfontfamily:NnnN #1 { BoldFont={},ItalicFont={},SmallCapsFont={},#2 } {#3} #4
  }
```

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

1.2 Font feature selection

`\@@_set_default_features:nn`

```
\cs_new:Nn \@@_set_default_features:nn
  {
    \IfBooleanTF {#1} \clist_gput_right:Nn \clist_gset:Nn
      \g_@@_default_fontopts_clist {#2}
  }
```

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

`\@@_set_font_default_features:nnn`

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
  {
⟨debug⟩ \typeout{\unexpanded{_set_font_default_features:nnn:{#1}{#2}{#3}}}
    \clist_map_inline:nn {#2}
      {
        \tl_if_single:nTF {##1}
          { \tl_set:No \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\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_gput:NVn    \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
          }
      }
  }
```

(*End of definition for* `\@@_set_font_default_features:nnn`. *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_new: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_@@_fontname_tl} }
          \use:x
            {
              \@@_select_font_family:nn
                { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
            }
        \group_end:
        \fontfamily \g_@@_nfss_family_tl \selectfont
      }
      {
        \@@_warning:nx {addfontfeatures-ignored} {#1}
      }
    \ignorespaces
  }
```

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

1.3 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_new:Nn \@@_main_newfontfeature:nn
  {
    \keys_define:nn { fontspec }
      {
        #1 .code:n = { \@@_update_featstr:n {#2} }
      }
  }
```

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

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

```
\cs_new: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 of definition for* `\aliasfontfeature`*. This function is documented on page* ??.)

`\aliasfontfeatureoption`

```
\cs_new: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 of definition for* `\aliasfontfeatureoption`*. This function is documented on page* ??.)

`\@@_main_DeclareFontExtensions:n`

```
\cs_new:Nn \@@_main_DeclareFontExtensions:n
  {
    \clist_set:Nn \l_@@_extensions_clist { #1 }
  }
```

Defaults:

```
\@@_main_DeclareFontExtensions:n {.otf,.ttf,.OTF,.TTF,.ttc,.TTC,.dfont}
```

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

1.4 High level conditionals

`\IfFontFeatureActiveTF`

```
\cs_new: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:n {#1}
    \group_end:

⟨debug⟩ \typeout{:::> \exp_not:N\g_@@_rawfeatures_sclist->~{\g_@@_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 of definition for* `\IfFontFeatureActiveTF`*. This function is documented on page* ??.)

1.5 `\oldstylenums` and `\liningnums`

`\oldstylenums`
`\liningnums`

This command needs a redefinition. And we may as well provide the reverse command.

```
\cs_new_protected:Nn \@@_main_oldstylenums:n
  {
    \group_begin:
      \addfontfeature{Numbers=OldStyle}
      #1
    \group_end:
  }
\cs_new_protected:Nn \@@_main_liningnums:n
  {
    \group_begin:
      \addfontfeature{Numbers=Lining}
      #1
    \group_end:
  }
```

(*End of definition for* `\oldstylenums` *and* `\liningnums`*. These functions are documented on page 27.*)

File IX

fontspec-code-api.dtx

1 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.)

2 Overview

2.1 Commands

`\fontspec_gset_family:Nnn`
`\fontspec_set_family:Nnn`

`\fontspec_set_family:Nnn` ⟨*family*⟩ `{`⟨*features*⟩`}` `{`⟨*font name*⟩`}`

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

`\fontspec_gset_fontface:NNnn`
`\fontspec_set_fontface:NNnn`

`\fontspec_set_fontface:NNnn` ⟨*face*⟩ ⟨*family*⟩ `{`⟨*features*⟩`}` `{`⟨*font name*⟩`}`

As for `\fontspec_set_family:Nnn` but with a single font face only. (E.g., no bold, italic shapes, etc.) The control sequence ⟨*face*⟩ is a primitive TEX font command.

2.2 Conditionals

`\fontspec_font_if_exist:nTF`

`\fontspec_font_if_exist:nTF {`⟨*font name*⟩`} Argtrue code {`⟨*false code*⟩`}`

Does this font exist? The font name can refer to the 'logical' name or to a filename with known font extension.

`\fontspec_if_fontspec_font:TF`

`\fontspec_if_fontspec_font:TF {`⟨*true code*⟩`} {`⟨*false code*⟩`}`

`\fontspec_if_aat_feature:nnTF`

`\fontspec_if_aat_feature:nnTF {`⟨*true code*⟩`} {`⟨*false code*⟩`}`

`\fontspec_if_opentype:TF`

`\fontspec_if_opentype:TF {`⟨*true code*⟩`} {`⟨*false code*⟩`}`

`\fontspec_if_feature:nTF`

```
\fontspec_if_feature:nTF {⟨feat tag⟩} {⟨true code⟩} {⟨false code⟩}
```

Check if the raw OpenType ⟨`feature tag`⟩ is available in the current font with script and language settings as set up when the font was loaded.

`\fontspec_if_feature:nnnTF`

```
\fontspec_if_feature:nnnTF {⟨script tag⟩} {⟨lang tag⟩} {⟨feat tag⟩} {⟨true code⟩}
{⟨false code⟩}
\fontspec_if_feature:nTF {latn} {ROM} {pnum} {True} {False}
```

Test whether the currently selected font with raw OpenType ⟨`script tag`⟩ and raw OpenType ⟨`language tag`⟩ contains the raw OpenType ⟨`feature tag`⟩. Returns false if the font is not loaded by fontspec or is not an OpenType font.

`\fontspec_if_script:nTF`

```
\fontspec_if_script:nTF {⟨script tag⟩} {⟨true code⟩} {⟨false code⟩}
\fontspec_if_script:nTF {latn} {True} {False}
```

Test whether the currently selected font contains the raw OpenType ⟨`script tag`⟩.

Returns false if the font is not loaded by fontspec or is not an OpenType font.

`\fontspec_if_language:nTF`

```
\fontspec_if_language:nTF {⟨lang tag⟩} {⟨true code⟩} {⟨false code⟩}
\fontspec_if_language:nTF {ROM} {True} {False}
```

Check if the raw OpenType ⟨`language tag`⟩ is available in the current font with script settings as set up when the font was loaded.

`\fontspec_if_language:nnTF`

```
\fontspec_if_language:nnTF {⟨script tag⟩} {⟨lang tag⟩} {⟨true code⟩} {⟨false code⟩}
\fontspec_if_language:nnTF {cyrl} {SRB} {True} {False}
```

Test whether the currently selected font contains the raw OpenType ⟨`language tag`⟩ in ⟨`script tag`⟩.

Returns false if the font is not loaded by fontspec or is not an OpenType font.

`\fontspec_if_current_script:nTF`

```
\fontspec_if_current_script:nTF {⟨script tag⟩} {⟨true code⟩} {⟨false code⟩}
```

Test whether the currently loaded font has been loaded with the specified raw OpenType ⟨`script tag`⟩.

`\fontspec_if_current_language:nTF`

```
\fontspec_if_current_language:nTF {⟨lang tag⟩} {⟨true code⟩} {⟨false code⟩}
```

Test whether the currently loaded font has been loaded with the specified raw OpenType ⟨`language tag`⟩.

`\fontspec_if_current_feature:nTF`

```
\fontspec_if_current_feature:nTF {⟨feat tag⟩} {⟨true code⟩} {⟨false code⟩}
```

Test whether the currently loaded font is using the specified raw OpenType ⟨`feature tag`⟩.

`\fontspec_if_small_caps:TF`

```
\fontspec_if_small_caps:TF {⟨true code⟩} {⟨false code⟩}
```

Test whether the current font has small caps available.

3 Implementation

`\fontspec_if_fontspec_font:TF`

```
\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 of definition for `\fontspec_if_fontspec_font:TF`*. This function is documented on page 41.)*

`\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 of definition for `\fontspec_if_aat_feature:nnTF`*. This function is documented on page 41.)*

`\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 of definition for `\fontspec_if_opentype:TF`*. This function is documented on page 41.)*

`\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_@@_script_tl
            \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag}    \l_@@_lang_tl

            \@@_check_ot_feat:NnnnTF \font {#1} {\l_@@_lang_tl} {\l_@@_script_tl} {\prg_return
          }
          {
            \prg_return_false:
          }
      }
      {
        \prg_return_false:
      }
  }
```

(*End of definition for* `\fontspec_if_feature:nTF`. *This function is documented on page 42.*)

`\fontspec_if_feature:nnnTF`

#1 : script tag
#2 : language tag
#3 : feature tag

```
\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
          {
            \@@_check_ot_feat:NnnnTF \font {#3} {#2} {#1} \prg_return_true: \prg_return_false:
          }
          { \prg_return_false: }
      }
      { \prg_return_false: }
  }
```

(*End of definition for* `\fontspec_if_feature:nnnTF`. *This function is documented on page 42.*)

`\fontspec_if_script:nTF`

#1 : script tag

```
\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 of definition for* `\fontspec_if_script:nTF`. *This function is documented on page 42.*)

`\fontspec_if_language:nTF` #1 : lang tag

```
\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_@@_tmp_tl
            \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
            \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag}  \l_@@_script_tl

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

(*End of definition for* `\fontspec_if_language:nTF`. *This function is documented on page 42.*)

`\fontspec_if_language:nnTF` #1 : script tag
#2 : lang tag

```
\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
          {
            \@@_check_lang:NnnTF \font {#2} {#1} \prg_return_true: \prg_return_false:
          }
          { \prg_return_false: }
      }
      { \prg_return_false: }
  }
```

(*End of definition for* `\fontspec_if_language:nnTF`. *This function is documented on page 42.*)

`\fontspec_if_current_script:nTF` #1 : script tag

```
\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 of definition for* `\fontspec_if_current_script:nTF`. *This function is documented on page 42.*)

`\fontspec_if_current_language:nTF` #1 : lang tag

```
\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 of definition for* `\fontspec_if_current_language:nTF`. *This function is documented on page 42.*)

`\fontspec_gset_family:Nnn`
`\fontspec_set_family:Nnn`

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

```
\cs_new:Nn \@@_tl_new_if_free:N { \tl_if_exist:NF #1 { \tl_new:N #1 } }
\cs_new:Nn \@@_set_family:NnnN
  {
⟨debug⟩\typeout{::::::~fontspec_set_family:Nnn}
    \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
    \tl_set:Nn \l_@@_family_label_tl {#1}
    \@@_select_font_family:nn {#2} {#3}
    \@@_tl_new_if_free:N #1
    #4 #1 \l_fontspec_family_tl
    \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
⟨debug⟩\typeout{::::::~END~fontspec_set_family:Nnn}
  }
\cs_new:Nn \fontspec_gset_family:Nnn { \@@_set_family:NnnN #1 {#2} {#3} \tl_gset_eq:NN }
\cs_new:Nn \fontspec_set_family:Nnn  { \@@_set_family:NnnN #1 {#2} {#3} \tl_set_eq:NN  }
```

```
\cs_generate_variant:Nn \fontspec_set_family:Nnn {c}
```

(End of definition for `\fontspec_gset_family:Nnn` *and* `\fontspec_set_family:Nnn`*. These functions are documented on page 41.)*

`\fontspec_gset_fontface:NNnn`
`\fontspec_set_fontface:NNnn`

TODO: the round-about approach of using `\fontname` means that settings such as fontdimens will be lost. (Discovered in unicode-math.) Investigate!

```
\tl_new:N \l_@@_fontface_cs_tl
\tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font}
\cs_new:Nn \@@_set_fontface:NNnnN
  {
    \tl_set:Nn \l_@@_fontface_cs_tl {#1}
    \tl_set:Nn \l_@@_family_label_tl {#2}
    \@@_select_font_family:nn {#3} {#4}
    #5 #2 \l_fontspec_family_tl
    \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
  }
\cs_new:Nn \fontspec_gset_fontface:NNnn { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_gset_eq:N
\cs_new:Nn \fontspec_set_fontface:NNnn  { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_set_eq:NN
```

(End of definition for `\fontspec_gset_fontface:NNnn` *and* `\fontspec_set_fontface:NNnn`*. These functions are documented on page 41.)*

`\fontspec_font_if_exist:nTF`

```
\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 of definition for `\fontspec_font_if_exist:nTF`*. This function is documented on page 41.)*

`\fontspec_if_current_feature:nTF`

`#1` : feat tag

```
\prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
  {
⟨debug⟩\typeout{::~fontspec_if_current_feature:n~{#1}}
⟨debug⟩\typeout{::::~primitive_font_current_name:~=~\@@_primitive_font_current_name:}
    \exp_args:Nxx \tl_if_in:nnTF
      { \@@_primitive_font_current_name: } { \tl_to_str:n {#1} }
      { \prg_return_true: } { \prg_return_false: }
  }
```

(End of definition for `\fontspec_if_current_feature:nTF`*. This function is documented on page 42.)*

\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/\shapedefault }
          { \prg_return_false: }
          { \prg_return_true:  }
      }
      { \prg_return_false: }
  }
```

(*End of definition for* `\fontspec_if_small_caps:TF`*. This function is documented on page 42.*)

File X

fontspec-code-internal.dtx

1 Internals

1.1 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}
    \tl_set_eq:NN \l_@@_fontname_up_tl \l_fontspec_fontname_tl
    \tl_set_eq:NN \l_@@_basename_tl    \l_fontspec_fontname_tl

⟨debug⟩\typeout{^^J::::::::::::::: l_fontspec_fontname_tl~ =~ \l_fontspec_fontname_tl }

    \@@_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:N \l_fontspec_fontname_tl

    \@@_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:

⟨debug⟩\typeout{^^J::::::::::::::: l_fontspec_fontname_tl~ =~ \l_fontspec_fontname_tl }
⟨debug⟩\typeout{::::::::::::::: _fontname_up_tl~ =~ \l_@@_fontname_up_tl }
⟨debug⟩\typeout{::::::::::::::: l_@@_extension_tl~ =~ \l_@@_extension_tl }

    \@@_load_font:
    \@@_set_scriptlang:
    \@@_get_features:n {}
    \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:

    \tl_set_eq:NN \l_fontspec_family_tl \g_@@_nfss_family_tl
⟨debug⟩\typeout{::::::::::::::::::::::::::::::}
  }
```

(*End of 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 of 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.

```
\cs_new:Nn \@@_sanitise_fontname:Nn
  {
    \tl_set:Nx #1 {#2}
    \tl_trim_spaces:N #1
    \@@_process_ext:N #1
  }

\cs_new:Nn \@@_process_ext:N
  {
    \clist_map_inline:Nn \l_@@_extensions_clist
      {
        \tl_if_in:NnT #1 {##1}
          {
⟨debug⟩  \typeout{::@@_process_ext:N~ ---~ Removing~ EXT:~ ##1}
            \tl_remove_once:Nn #1 {##1}
            \tl_set:Nn \l_@@_extension_tl {##1}
            \@@_font_is_file:
```

```
            \clist_map_break:
          }
      }
  }
```

(*End of 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 of 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:eeT { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
      {
        \tl_set_eq:NN \l_@@_fontname_it_tl   \l_fontspec_fontname_tl
        \tl_set_eq:NN \l_@@_fontname_bf_tl   \l_fontspec_fontname_tl
        \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
      }
  }
```

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

`\@@_load_external_fontoptions:N`

Load a possible `.fontspec` font configuration file. This file could set font-specific options for the font about to be loaded. The parameter should be a token list containing a sanitised fontname. In the past this used a space-stripped version of the name, so we check for the file both with and without spaces to load it.

```
\cs_new:Nn \@@_load_external_fontoptions:N
  {
    \bool_if:NT \l_@@_fontcfg_bool
      {
⟨debug⟩  \typeout{:: @@_load_external_fontoptions:N \exp_not:N #1 }
        \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:nTF { \l_@@_ext_filename_tl }
            {
              \file_input:n { \l_@@_ext_filename_tl }
            }
            {
              \tl_remove_all:Nn \l_@@_ext_filename_tl {~}
              \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
                { \file_input:n { \l_@@_ext_filename_tl } }
            }
         }
      }
  }
```

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

`\@@_load_font:`

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

    \@@_sanitise_fontname:Nn \l_@@_fontname_up_tl { \l_@@_fontname_up_tl }
⟨debug⟩\typeout{Set~ base~ font~ for~ preliminary~ analysis:~ "\l_@@_fontname_up_tl"~ with~ fe
    \@@_primitive_font_set:NnnF \l_@@_test_font
      { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
      { \f@size pt - 2sp }
      { \@@_error:nx {font-not-found} {\l_@@_fontname_up_tl} }

⟨debug⟩\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
    \@@_set_font_type:N \l_@@_test_font
    \@@_primitive_font_gset:Onn \l_@@_fontface_cs_tl
      {  \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
      { \f@size pt + 2sp }

    \l_@@_fontface_cs_tl % this is necessary for LuaLaTeX to check the scripts properly

  }
```

(*End of 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_new:Nn \@@_construct_font_call:nnnnnn
  {
```

```
⟨XE⟩   " \@@_fontname_wrap:n { #1 #2 #3 }
⟨LU⟩   " \@@_fontname_wrap:n { #1 #2 } #3
      #4 #5
      \str_if_eq:eeF {#6}{} {:#6} "
  }
```

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

```
\cs_new:Nn \@@_construct_font_call:nn
  {
    \@@_construct_font_call:nnnnnn
      {#1}
      \l_@@_extension_tl
      \l_@@_ttc_index_tl
      \l_@@_renderer_tl
      \l_@@_optical_size_tl
      {#2}
  }
```

(*End of 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'. For LuaTEX there are two kinds kinds of filename based loading supported: Regular filename lookups which include system fonts and lookups restricted to kpse.

```
\cs_new:Nn \@@_font_is_name:
  {
⟨XE⟩   \cs_set_eq:NN \@@_fontname_wrap:n \use:n
⟨LU⟩   \cs_set:Npn \@@_fontname_wrap:n ##1 { name: ##1 }
  }
```

```
\cs_new:Nn \@@_font_is_file:
  {
⟨debug⟩   \typeout{:: _font_is_file:}
    \bool_set_true:N \l_@@_external_bool
    \bool_lazy_and:nnTF { \l_@@_external_kpse_bool } { \tl_if_empty_p:N \l_@@_font_path_tl }
      {
        \cs_set:Npn \@@_fontname_wrap:n ##1 { kpse: ##1 }
      }
      {
        \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
      }
  }
```

(*End of 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:
  {
⟨debug⟩   \typeout{:: _set_scriptlang:}
    \bool_if:NT \l_@@_firsttime_bool
      {
        \tl_if_empty:NF \l_@@_script_name_tl
          {
⟨debug⟩  \typeout{:::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
            \keys_set:ne {fontspec-opentype} {Script=\l_@@_script_name_tl}
            \keys_set:ne {fontspec-opentype} {Language=\l_@@_lang_name_tl}
          }
      }
  }
```

(*End of 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_new:Nn \@@_get_features:n
  {
⟨debug⟩  \typeout{:: @@_get_features:Nn { \exp_not:n {#1} } }
    \@@_init_fontface:
    \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
      \l_@@_keys_leftover_clist
    \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
⟨*XE⟩
    \bool_if:NTF \l_@@_ot_bool
      {
⟨debug⟩  \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
        \keys_set_known:nV {fontspec-opentype} \l_@@_keys_leftover_clist
      }
      {
⟨debug⟩  \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features:~"\l_@@_keys_leftover_c
        \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
          { \keys_set_known:nV {fontspec-aat} \l_@@_keys_leftover_clist }
      }
⟨/XE⟩
⟨*LU⟩
⟨debug⟩  \typeout{::: Setting~ keys~ for~ OpenType~ font~ features:~"\l_@@_keys_leftover_clist
    \keys_set_known:nV {fontspec-opentype} \l_@@_keys_leftover_clist
⟨/LU⟩

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

    \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
                     { \c_@@_hexcol_tl \c_@@_opacity_tl }
⟨XE⟩       { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
⟨LU⟩       { \@@_update_featstr:n { color = {\l_@@_hexcol_tl\l_@@_opacity_tl} } }
  }
```

(*End of 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_empty:NTF \l_@@_nfss_fam_tl
    {
      \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
        {
          \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
          \prg_return_false:
        }
        {
          \tl_set:Nx \l_@@_tmp_tl {#1}
          \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
          \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
          \prg_return_true:
        }
    }
    {
      \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
      \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
      \prg_return_true:
    }
  }
```

```
\cs_new:Nn \@@_save_fontid_family:nn
  {
    \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
      {
        \tl_set:Nx \l_@@_tmp_tl
          { \int_eval:n { \l_@@_tmp_tl + 1 } }
      }
      { \tl_set:Nn \l_@@_tmp_tl { 0 } }
    \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
    \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
    \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
  }
\cs_generate_variant:Nn \@@_save_fontid_family:nn { VV }
```

(*End of 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{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{}
    \@@_set_faces:
    \@@_info:nxx {defining-font} {#1} {#2}
  }
```

(*End of 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_ \g_@@_nfss_family_tl _prop}
  \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
  \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options}  { \l_@@_all_features
  \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
   {
    \@@_construct_font_call:nn {\l_fontspec_fontname_tl}
     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist \@@_get_variations: }
   }
  \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
  \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num}   \l_@@_language_int
  \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl
  \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag}   \l_@@_lang_tl
 }
```

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

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

`\@@_set_faces:`

```
\cs_new:Nn \@@_set_faces:
  {
    \@@_add_nfssfont:nnnn \mddefault \shapedefault \l_fontspec_fontname_tl \l_@@_fontfeat_up_c
    \@@_add_nfssfont:nnnn \bfdefault \shapedefault \l_@@_fontname_bf_tl    \l_@@_fontfeat_bf_c
    \@@_add_nfssfont:nnnn \mddefault \itdefault    \l_@@_fontname_it_tl    \l_@@_fontfeat_it_c
    \@@_add_nfssfont:nnnn \mddefault \sldefault    \l_@@_fontname_sl_tl    \l_@@_fontfeat_sl_c
    \@@_add_nfssfont:nnnn \mddefault \swdefault    \l_@@_fontname_sw_tl    \l_@@_fontfeat_sw_c
    \@@_add_nfssfont:nnnn \bfdefault \itdefault    \l_@@_fontname_bfit_tl  \l_@@_fontfeat_bfit
    \@@_add_nfssfont:nnnn \bfdefault \sldefault    \l_@@_fontname_bfsl_tl  \l_@@_fontfeat_bfsl
    \@@_add_nfssfont:nnnn \bfdefault \swdefault    \l_@@_fontname_bfsw_tl  \l_@@_fontfeat_bfsw
    \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nnnnn ##2 }
  }
\cs_new:Nn \@@_set_faces_aux:nnnnn
  {
⟨debug⟩  \typeout{:: @@_set_faces_aux:nnnnn  \exp_not:n { {#1} {#2} {#3} {#4} {#5} } }
    \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
    \@@_make_font_shapes:Nnnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
  }
```

(*End of 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_new:Nn \fontspec_complete_fontname:Nn
  {
    \tl_set:Nx #1 {#2}
    \tl_if_in:NnF \l_fontspec_fontname_tl {*}
      {
        \tl_replace_all:Nne #1 {*} {\l_@@_basename_tl}
      }
  }
```

(*End of 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:eTF {#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:Nee \l_@@_nfssfont_prop {#1/#2}
          { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
      }
  }
```

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

1.2.1 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_test_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:}
⟨*XE⟩
  \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
⟨debug⟩  \typeout{:::: TFM}
    \bool_set_true:N \l_@@_tfm_bool
  \or
⟨debug⟩  \typeout{:::: AAT}
    \bool_set_true:N \l_@@_atsui_bool
    \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/AAT} }
    \ifnum\XeTeXcountvariations #1 > 0\relax
⟨debug⟩  \typeout{:::: MM}
      \bool_set_true:N \l_@@_mm_bool
    \fi
  \or
⟨debug⟩  \typeout{:::: OpenType}
    \bool_set_true:N \l_@@_ot_bool
    \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/OT} }
  \or
⟨debug⟩  \typeout{:::: Graphite}
    \bool_set_true:N \l_@@_graphite_bool
    \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/GR} }
  \fi
⟨/XE⟩
```

If automatic, the `\l_@@_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:

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

(End of 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 XƎTEX 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:eF {#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}
  {
    \group_begin:
    \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1}   { \l_@@_pre
    \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} { \l_@@_pre
    \cs_if_eq:NNTF \l_@@_tmpa_font \l_@@_tmpb_font
      { \group_end: \prg_return_false: }
      { \group_end: \prg_return_true: }
  }
```

(End of 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:Nn \l_fontspec_fontname_tl {#1}
      \@@_declare_shape:nnxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
    \group_end:
  }
\cs_new:Nn \@@_load_fontname:Nn
  {
⟨debug⟩    \typeout{:: @@_load_fontname:Nn \exp_not:N #1 (#1) {#2} }
    \@@_sanitise_fontname:Nn #1 {#2}
    \@@_load_external_fontoptions:N #1
    \prop_get:NVNF \g_@@_fontopts_prop #1 \l_@@_fontopts_clist
      { \clist_clear:N \l_@@_fontopts_clist }
    \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
    \@@_primitive_font_set:OnnF \l_@@_fontface_cs_tl
      { \@@_construct_font_call:nn {#1} { \l_@@_pre_feat_sclist } } { \f@size pt + 2sp }
      { \@@_error:nx {font-not-found} {#2} }
  }
\keys_define:nn {fontspec/fontname}
  {
    Font .tl_set:N = \l_fontspec_fontname_tl ,
    Font .groups:n = {getfontname} ,
  }
```

(*End of 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_build_begin:N \l_@@_nfss_tl
    \tl_build_begin: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} }

    \tl_build_end:N \l_@@_nfss_tl
    \tl_build_end:N \l_@@_nfss_sc_tl

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

(End of 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:neN {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:Nn \l_fontspec_fontname_tl {\l_@@_sizedfont_tl}
    \@@_setup_nfss:Nn \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:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl} }% local for e
      }

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

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

`\@@_setup_nfss:Nn`

```
\cs_new:Nn \@@_setup_nfss:Nn
  {
⟨debug⟩\typeout{====~Setup~NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
⟨debug⟩\typeout{====~Requested~features:~#2}

    \@@_get_features:n { #2 }

⟨debug⟩\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist \@@_get_variations:}

    \tl_if_empty:NF \l_@@_scale_tl
      {
        \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
      }

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

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

`\@@_declare_shapes_normal:nn`

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

(*End of 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 {\g_@@_nfss_enc_tl} {\g_@@_nfss_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#1 }
  }
```

(*End of 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_p:ee {#2} {\itdefault}  &&
        !(\str_if_eq_p:ee {\itdefault} {\sldefault})
      }
      {
        \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
          {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
      }
  }
```

`\@@_declare_shapes_bx:nn`

Similar processing for setting up b/bx substitutions.

```
\cs_new:Nn \@@_declare_shapes_bx:nn
  {
    \bool_if:nT
      {
          \str_if_eq_p:ee {#1} {\bfdefault}  &&
        !(\str_if_eq_p:ee {\bfdefault} {bx})
      }
      {
        % bx/?
        \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
```

```
     {bx} {#2}
     { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
     { \l_@@_postadjust_tl }

    % bx/sc -> b/sc
    \tl_if_empty:NF \l_@@_nfss_sc_tl
     {
      \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
       {bx} { \@@_combo_sc_shape:n {#2} }
       { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
       { \l_@@_postadjust_tl }
     }

    % bx/sl -> bx/it
    \bool_if:nT
     {
       \str_if_eq_p:ee {#2} {\itdefault}  &&
      !(\str_if_eq_p:ee {\itdefault} {\sldefault})
     }
     {
      \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
       {bx} {\sldefault}
       { <->ssub*\g_@@_nfss_family_tl/bx/\itdefault }
       { \l_@@_postadjust_tl }
     }

   }
 }
```

Lastly some informative messaging.

`\@@_declare_shape_loginfo:nn`

```
\cs_new:Nn \@@_declare_shape_loginfo:nn
 {
  \tl_gput_right:Nx \g_@@_defined_shapes_tl
   {
    \exp_not:n { \\ }
    -~ \exp_not:N \str_case:nn {#1/#2}
    {
     {\mddefault/\shapedefault} {'normal'~}
     {\bfdefault/\shapedefault} {'bold'~}
     {\mddefault/\itdefault} {'italic'~}
     {\mddefault/\sldefault} {'slanted'~}
     {\mddefault/\swdefault} {'swash'~}
     {\bfdefault/\itdefault} {'bold~ italic'~}
     {\bfdefault/\sldefault} {'bold~ slanted'~}
     {\bfdefault/\swdefault} {'bold~ swash'~}
    } (#1/#2)~
    with~ NFSS~ spec.:~
    \l_@@_nfss_tl
    \tl_if_empty:NF \l_@@_nfss_sc_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/\scitdefault} {'italic~ small~ caps'~}
              {\bfdefault/\scitdefault} {'bold~ italic~ small~ caps'~}
              {\mddefault/\scsldefault} {'slanted~ small~ caps'~}
              {\bfdefault/\scsldefault} {'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:eeF` would be better?

1.2.2 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
⟨*XE⟩
  {
    \bool_if:NT \l_@@_ot_bool
      {
        \tl_if_empty:NF \l_@@_script_tl { script   = \l_@@_script_tl ; }
        \tl_if_empty:NF \l_@@_lang_tl   { language = \l_@@_lang_tl   ; }
      }
  }
⟨/XE⟩
⟨*LU⟩
  {
    mode     = \l_@@_mode_tl   ;
    \tl_if_empty:NF \l_@@_shaper_tl { shaper = \l_@@_shaper_tl   ; }
    \tl_if_empty:NF \l_@@_script_tl { script   = \l_@@_script_tl ; }
    \tl_if_empty:NF \l_@@_lang_tl   { language = \l_@@_lang_tl   ; }
  }
⟨/LU⟩
```

This macro checks if the font contains small caps.

`\@@_make_smallcaps:TF`
`\@@_make_ot_smallcaps:TF`

```
\cs_new:Nn \@@_make_ot_smallcaps:TF
  {
    \bool_set_false:N \l_@@_tmpa_bool
    \exp_args:Ne \clist_map_inline:nn { \l_@@_lang_tl , \g_@@_default_langs_clist }
      {
        \exp_args:Ne \clist_map_inline:nn { \l_@@_script_tl , \g_@@_default_scripts_clist }
          {
            \exp_args:No \@@_check_ot_feat:NnnnT \l_@@_fontface_cs_tl {smcp} {##1} {####1}
```

```
       {
⟨debug⟩\typeout{SMCP~found~for~script/lang: ####1/##1~--~assuming~okay}
        \bool_set_true:N \l_@@_tmpa_bool
        \clist_map_break:
       }
     }
   }
  \bool_if:NTF \l_@@_tmpa_bool {#1} {#2}
 }
\cs_new:Nn \@@_make_smallcaps:TF
 {
  \bool_if:NTF \l_@@_ot_bool
   { \@@_make_ot_smallcaps:TF {#1} {#2} }
   {
    \bool_if:NT \l_@@_atsui_bool
     {
      \exp_args:No \@@_make_AAT_feature_string:NnnTF
       \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
     }
   }
 }
```

\@@_update_featstr:n

`\g_@@_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  \g_@@_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 \g_@@_rawfeatures_sclist {##1;}
   }
 }
\cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}
```

\@@_get_variations:

`\@@_get_variations:` builds the feature string representing the current variation instance and/or axis settings.

```
\cs_generate_variant:Nn \tl_tail:n { e }
```

```
\cs_new:Nn \@@_format_axis:nn
  {
    , #1 = #2
  }
\cs_new:Nn \@@_get_variations:
  {
    \tl_if_empty:NF \g_@@_instance_tl
      {
        instance = { \g_@@_instance_tl };
      }
      \prop_if_empty:NF \g_@@_rawvariations_prop
      {
        axis = {
          \tl_tail:e {
            \prop_map_function:NN \g_@@_rawvariations_prop \@@_format_axis:nn
          }
        };
      }
  }
```

1.3 Initialisation

`\@@_init:`

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.)

```
\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_@@_renderer_tl
    \tl_gclear:N \g_@@_defined_shapes_tl
    \tl_gclear:N \g_@@_curr_series_tl
    \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspec_encoding_tl
⟨*LU⟩
    \tl_set:Nn \l_@@_mode_tl {node}
⟨/LU⟩
  }
```

Executed in `\@@_get_features:Nn`.

`\@@_init_fontface:`

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

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

`\@@_ot_validate_tag:n`

```
⟨*LU⟩
\cs_new_protected:Nn \@@_ot_validate_tag:n
  {
    \@@_ot_validate_tag:w #1 \q_nil
  }
\cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
\cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
  {
    \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
      { \@@_ot_validate_tag_aux:w #2   \c_empty_tl \c_empty_tl \q_nil }
      { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
  }
\cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
  {
    \int_compare:nT { \tl_count:n {#5} > 2 }
      { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
  }
⟨/LU⟩
```

`\@@_iv_str_to_num:Nn`

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

This code is not used in LuaTEX, as the checking for that engine is done via Lua code provided by luaotfload.

```
⟨*XE⟩
\cs_new:Nn \@@_iv_str_to_num:Nn
  {
⟨debug⟩\typeout{_iv_str_to_num:~#1~/~#2}
    \@@_strip_leading_sign:Nw #1#2 \q_nil
  }
\cs_generate_variant:Nn \@@_iv_str_to_num:Nn {Nx}
```

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 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: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 }
 }
```

If input string (after sign is stripped) is more than 4 chars, `#6` will contain '⟨*excess*⟩`\c_empty_tl\c_empty_tl`'. Therefore use `#6` to verify string length.

```
\cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
 {
  \int_compare:nT { \tl_count:n {#6} > 2 }
   { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }

  \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
   }
 }
⟨/XE⟩
```

File XI

fontspec-code-opentype.dtx

1 OpenType definitions code

\@@_define_opentype_variation_axis:nn

```
\cs_new:Nn \@@_define_opentype_variation_axis:nn
  {
    \keys_define:nn {fontspec-opentype}
      {
        #1 .code:n = {
          \prop_gput:Nnn \g_@@_rawvariations_prop { #2 } { ##1 }
        },
        #1 .value_required:n = true,
        #1 .groups:n = {opentype},
      }
  }
```

\@@_define_opentype_feature_group:n

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

\@@_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} } ,
       #1/#2 .groups:n = {opentype}
      }
    }
    {
     \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}
        } ,
       #1/#2 .groups:n = {opentype}
      }
    }
 }
```

\@@_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}
 }
```

1.1 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} } }
    \@@_remove_clashing_featstr:x { #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 of 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. \l_@@_script_int is used to store the number corresponding to the script tag string.

```
\prg_new_conditional:Nnn \@@_check_script:Nn {TF,T,F}
  {
⟨debug⟩\typeout{:: _check_script:Nn~#1~/~#2}
    \bool_if:NTF \l_@@_never_check_bool
      { \prg_return_true: }
      {
    \bool_if:nTF { \tl_if_empty_p:e {#2} }
      { \prg_return_false: }
      {
⟨*XE⟩
⟨debug⟩\typeout{::::~ checking~ script~ #2}
        \@@_iv_str_to_num:Nx \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:
⟨/XE⟩
⟨*LU⟩
        \@@_ot_validate_tag:x {#2}
        \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} }
⟨debug⟩\typeout{::::~ checking:~"\l_@@_tmp_tl",~ "#2"}
        \lua_now:e { fontspec.check_ot_script("\l_@@_tmp_tl", "#2") }
        \bool_if:NTF \l__fontspec_check_bool
          {
⟨debug⟩\typeout{:::::~ TRUE}
```

```
            \prg_return_true:
          }
          {
⟨debug⟩\typeout{::::::~ FALSE}
            \prg_return_false:
          }
⟨/LU⟩
      }
      }
  }
```

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

`\@@_check_lang:NnnTF`
`\@@_check_lang:NnTF`

This macro takes an OpenType language tag and checks if it exists in the current font/script. `\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,F}
  {
    \@@_check_lang:NnnTF #1 {#2} {\l_@@_script_tl} {\prg_return_true:} {\prg_return_false:}
  }
```

```
\prg_new_conditional:Nnn \@@_check_lang:Nnn {TF}
  {
⟨debug⟩\typeout{:: _check_lang:Nn~#1~/~#2~/~#3~/}
    \bool_if:NTF \l_@@_never_check_bool
      { \prg_return_true: }
      {
    \bool_if:nTF { \tl_if_empty_p:e {#3} }
      { \prg_return_false: }
      {
⟨*XE⟩
        \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
        \@@_iv_str_to_num:Nx \l_@@_script_int {#3}
        \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 }
          {
            \int_set:Nn \l_@@_tmpc_int
              { \XeTeXOTlanguagetag #1 \l_@@_script_int \l_@@_tmpa_int }

            \int_compare:nNnTF \l_@@_tmpc_int = \l_@@_strnum_int
              {
                \bool_set_true:N \l__fontspec_check_bool
                \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
              }
              {
                \int_incr:N \l_@@_tmpa_int
              }
          }
        \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
⟨/XE⟩
⟨*LU⟩
        \@@_ot_validate_tag:x {#2}
        \@@_ot_validate_tag:x {#3}
        \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} }
        \@@_lua_function:neee {check_ot_lang} {\l_@@_tmp_tl} {#2} {#3}
        \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
⟨/LU⟩
      }
      }
  }
```

(*End of definition for* `\@@_check_lang:NnnTF` *and* `\@@_check_lang:NnTF`*. These functions are documented on page* ??.)

`\@@_check_ot_feat:NnnnTF`

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:Nnnn {TF,T,F}
  {
    \bool_if:NTF \l_@@_never_check_bool
      { \prg_return_true: }
      {
    \bool_if:nTF { \tl_if_empty_p:e {#3} || \tl_if_empty_p:e {#4} }
      { \prg_return_false: }
      {
⟨*XE⟩
⟨debug⟩\typeout{::~ fontspec_check_ot_feat:nnn~ {#2}{#3}{#4}}
        \@@_iv_str_to_num:Nx \l_@@_strnum_int   {#2}

        \str_if_eq:eeTF {#3} {dflt}
          { \int_zero:N \l_@@_language_int }
          { \@@_iv_str_to_num:Nx \l_@@_language_int {#3} }
        \@@_iv_str_to_num:Nx \l_@@_script_int   {#4}

        \int_set:Nn \l_tmpb_int
          { \XeTeXOTcountfeatures #1 \l_@@_script_int \l_@@_language_int }

        \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:
⟨/XE⟩
⟨*LU⟩
⟨debug⟩\typeout{::~ fontspec_check_ot_feat:n~ {#1}}
        \@@_ot_validate_tag:x {#2}
        \@@_ot_validate_tag:x {#3}
        \@@_ot_validate_tag:x {#4}
        \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} }
        \@@_lua_function:neeee {check_ot_feat} {\l_@@_tmp_tl} {#2} {#3} {#4}
        \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
⟨/LU⟩
      }
      }
  }
```

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

1.2 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~/~Decomposition}
\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~Substitution}
\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~Me
\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}
```

TODO: move the above elsewhere!!

File XII

fontspec-code-graphite.dtx

1 Graphite/AAT code

`\@@_define_aat_feature_group:n`

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

(*End of 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 of 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} }
      {
        \exp_args:No \@@_make_AAT_feature_string:NnnTF \l_@@_fontface_cs_tl {#1} {#2}
          {
            \@@_update_featstr:n {\l_fontspec_feature_string_tl}
          }
          {
            \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2}
          }
      }
  }
```

(*End of 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 66).

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

File XIII

fontspec-code-keyval.dtx

1 Font loading (keyval) definitions

This package uses a large number of keyval modules which operate sequentially on keyval input to ensure priority.

```
\clist_gset:Nn \g_@@_all_keyval_modules_clist
  {
    fontspec, fontspec-opentype, fontspec-aat,
    fontspec-preparse, fontspec-preparse-cfg, fontspec-preparse-external, fontspec-preparse-ne
    fontspec-renderer
  }
```

Wrapper function to save some characters in the source:

```
\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} }
  }
```

1.1 Pre-pre-parsing stages

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
    \tl_set:Nn \l_@@_font_path_tl {#1}
    \@@_font_is_file:
⟨*XE⟩
    \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
⟨/XE⟩
  }
\aliasfontfeature{Path}{ExternalLocation}
\@@_keys_define_code:nnn {fontspec} {Path} {}
```

(*End of 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} {}
```

`KpseOnly` If the font is specified by filename, only search for it through kpse. XƎTEX does not support finding system fonts by filename so this is always implicitly set there.

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {KpseOnly}
  {
    \bool_set_true:N \l_@@_external_kpse_bool
    \bool_if:NT \l_@@_external_bool \@@_font_is_file:
  }
\@@_keys_define_code:nnn {fontspec} {KpseOnly} {}
```

`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 whether certain features are available.

```
⟨*XE⟩
\keys_define:nn {fontspec-renderer}
  {
    Renderer .choices:nn =
      {AAT,ICU,OpenType,Graphite,Full,Basic,Node,Base,HarfBuzz,Harfbuzz}
      {
        \int_compare:nTF {\l_keys_choice_int <= 4}
          {
            \tl_set:Nx \l_@@_renderer_tl
              {
                \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
              }
⟨debug⟩\typeout{Renderer:~ \l_@@_renderer_tl}
            \tl_gset:Nx \g_@@_single_feat_tl { \l_@@_renderer_tl }
          }
          {
            \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic/Node/Base/HarfBuzz}
          }
      }
  }
⟨/XE⟩
⟨*LU⟩
\keys_define:nn {fontspec-renderer}
  {
    Renderer .choices:nn =
      {Full,Node,Basic,Base,HarfBuzz,Harfbuzz,OpenType,AAT,Graphite}
      {
        \int_compare:nT {\l_keys_choice_int >= 5} { \bool_set_true:N \l_@@_harfbuzz_bool }

        \tl_set:Nx \l_@@_mode_tl
          {
            \int_case:nn \l_keys_choice_int { 1 {node} 2 {node} 3 {base} 4 {base} 5 {harf} 6 {
          }

        \tl_set:Nx \l_@@_shaper_tl
          {
            \int_case:nn \l_keys_choice_int { 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {ot} 8 {coretext
          }

⟨debug⟩\typeout{Mode:~"\l_@@_mode_tl"~/~Shaper:~"\l_@@_shaper_tl"}

        \tl_gset:Nx \g_@@_single_feat_tl
          {
            mode=\l_@@_mode_tl ;
            \tl_if_empty:NF \l_@@_shaper_tl { shaper=\l_@@_shaper_tl}
          }
      } ,

    Renderer unknown .code:n =
      {
        \bool_set_true:N \l_@@_harfbuzz_bool
        \@@_warning:nx {unknown-renderer} {#1}
        \tl_set:Nn \l_@@_mode_tl {harf}
        \tl_set:Nn \l_@@_shaper_tl {#1}
      } ,
  }
⟨/LU⟩
```

1.2 Pre-parsed features

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

```
\@@_keys_define_code:nnn {fontspec-preparse} {Script}
  {
⟨XE⟩    \tl_if_empty:NT \l_@@_renderer_tl { \keys_set:nn {fontspec-renderer} {Renderer=OpenTyp
    \tl_set:Nn \l_@@_script_name_tl {#1}
  }
```

Exactly the same:

```
\@@_keys_define_code:nnn {fontspec-preparse} {Language}
  {
⟨XE⟩    \tl_if_empty:NT \l_@@_renderer_tl { \keys_set:nn {fontspec-renderer} {Renderer=OpenTyp
    \tl_set:Nn \l_@@_lang_name_tl {#1}
  }
```

TTC font index

```
\@@_keys_define_code:nnn {fontspec-preparse} {FontIndex}
  {
    \str_if_eq:eeF { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
      { \@@_warning:n {font-index-needs-ttc} }
⟨XE⟩  \tl_set:Nn \l_@@_ttc_index_tl {:#1}
⟨LU⟩  \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
  }
\@@_keys_define_code:nnn {fontspec} {FontIndex}
  {
⟨XE⟩  \tl_set:Nn \l_@@_ttc_index_tl {:#1}
⟨LU⟩  \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
  }
```

1.3 Font faces

Upright

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

Italic and slanted

```
\@@_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}
      }
  }
```

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

```
\@@_keys_define_code:nnn {fontspec-preparse-external} {SwashFont}
  {
    \fontspec_complete_fontname:Nn \l_@@_fontname_sw_tl {#1}
  }
```

Bold (NFSS) Series By default, fontspec uses the default bold series, `\bfdefault`. We want to be able to make this extensible. This code is not yet functional!

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

Bold This contains some stubb code to allow more than one bold font to be loaded.

```
\@@_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 \l_@@_bf_series_seq
          {
            \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
            \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
          }

        \tl_if_eq:oeT \g_@@_curr_series_tl {\bfdefault}
          {
            \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
          }

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

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


      }
  }
```

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} {BoldSlantedFont}
  {
    \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse-external} {BoldSwashFont}
  {
    \fontspec_complete_fontname:Nn \l_@@_fontname_bfsw_tl {#1}
  }
```

Small caps 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}
      }
  }
```

1.3.1 Preparsed font features

```
\@@_keys_define_code:nnn {fontspec-preparse} {UprightFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_up_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_bf_clist {#1}
%  \prop_put:NeV \l_@@_nfss_prop
%     {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
  }
\@@_keys_define_code:nnn {fontspec-preparse} {ItalicFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_it_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldItalicFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_bfit_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {SlantedFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_sl_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldSlantedFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_bfsl_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {SwashFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_sw_clist {#1}
  }
\@@_keys_define_code:nnn {fontspec-preparse} {BoldSwashFeatures}
  {
    \clist_put_right:Nn \l_@@_fontfeat_bfsw_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_put_right:Nn \l_@@_fontfeat_sc_clist {#1}
      }
  }
```

Features 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}
  }
```

A hack to fix a test, needs to be investigated why necessary!

```
\@@_keys_define_code:nnn {fontspec-opentype} {UprightFont} {}
\@@_keys_define_code:nnn {fontspec-opentype} {ItalicFont} {}
\@@_keys_define_code:nnn {fontspec-opentype} {SlantedFont} {}
\@@_keys_define_code:nnn {fontspec-opentype} {BoldFont} {}
\@@_keys_define_code:nnn {fontspec-opentype} {BoldItalicFont} {}
\@@_keys_define_code:nnn {fontspec-opentype} {BoldSlantedFont} {}
```

1.4 General 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 \g_@@_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 }
  }
```

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

```
\@@_keys_define_code:nnn {fontspec-preparse} {FontFace}
  {
    \tl_clear:N \l_@@_this_font_tl
    \clist_set:No \l_@@_arg_clist { \use_iii:nnn #1 }
    \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
    \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
      {
⟨debug⟩\typeout{FontFace~ parsing:~ one~ clist~ item}
        \tl_if_in:NnF \l_@@_arg_clist {=}
          {
⟨debug⟩\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
            \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
            \tl_clear:N \l_@@_this_feat_clist
          }
      }

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

Scale If the input isn't one of the pre-defined string options, then it's gotta be numerical. `\fontspec_calc_scale:n` and `\fontspec_calc_scale:nn` do 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} }
        {MatchAveragecase} { \@@_calc_scale:nn {5} {8} }
      }
      { \tl_set:Nx \l_@@_scale_tl {#1} }
      \@@_info:n {set-scale}
  }
```

ScaleAgain

```
\@@_keys_define_code:nnn {fontspec} {ScaleAgain}
  {
    \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
    \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
    \@@_info:n {set-scale}
  }
```

`\@@_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 XeTeX).

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 { \familydefault }
      \selectfont

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

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

      \exp_args:NNNx
    \group_end:
    \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
  }
```

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

`\@@_calc_scale:nn` This macro calls `\fontspec_calc_scale:n` twice and then sets the scale to the average of the two results.

```
\cs_new:Nn \@@_calc_scale:nn
{
    \group_begin:
      \__fontspec_calc_scale:n {#1}
      \tl_set_eq:NN \l_@@_tmp_tl \l_@@_scale_tl
      \__fontspec_calc_scale:n {#2}
      \tl_set:Nx \l_@@_scale_tl
    {
     \fp_eval:n { (\l_@@_tmp_tl + \l_@@_scale_tl)/2 }
    }
   \exp_args:NNNx
  \group_end:
  \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
 }
```

(*End of definition for* `\@@_calc_scale:nn`. *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 of 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 of 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_e: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} }
      }
      {
⟨LU⟩        \@@_warning:nx {only-xetex-feature} {HyphenChar}

        \tl_if_single:nTF {#1}
          { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
          { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }

        \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphen
          {
            \tl_put_right:Nn \l_@@_postadjust_tl
              { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
          }
          { \@@_error:nxx {no-glyph}{\l_fontspec_fontname_tl}{#1} }

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

Color Test first if the color is a named l3color, then if it is a color from xcolor, which names its colours `\color@<name>`. If this fails the argument is assumed to be a hex color.

```
\@@_keys_define_code:nnn {fontspec} {Color}
  {
⟨*XE⟩
    \color_if_exist:nTF {#1}
      {
        \color_export:nnN {#1} {HTML}\l_@@_hexcol_tl
      }
      {
        \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} }
                  }
              }
          }
      }
⟨/XE⟩
⟨*LU⟩
    \color_if_exist:nTF {#1}
      {
        \tl_set:Nn \l_@@_hexcol_tl {#1}
      }
      {
        \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} }
                  }
              }
          }
      }
⟨/LU⟩
  }

\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 \c_@@_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 \c_@@_opacity_tl
      {
        \bool_if:NF \l_@@_firsttime_bool
        { \@@_warning:nx {opa-twice} {#1} }
      }
    \tl_set:Nx \l_@@_opacity_tl
      {
⟨LU⟩      ,
        \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
        \int_to_hex:n { \l_@@_tmp_int }
      }
  }
```

Mapping

```
⟨*XE⟩
\@@_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 }
  }
⟨/XE⟩
⟨*LU⟩
\@@_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} }
  }
⟨/LU⟩
```

1.4.1 Continuous font axes

```
⟨*XE⟩
\@@_keys_define_code:nnn {fontspec} {Weight}
  {
    \@@_update_featstr:n{weight=#1}
  }
⟨/XE⟩
⟨LU⟩\@@_define_opentype_variation_axis:nn {Weight} {wght}
⟨*XE⟩
\@@_keys_define_code:nnn {fontspec} {Width}
  {
    \@@_update_featstr:n{width=#1}
  }
⟨/XE⟩
⟨LU⟩\@@_define_opentype_variation_axis:nn {Width} {wdth}
\@@_define_opentype_variation_axis:nn {Slant} {slnt}
```

```
\@@_keys_define_code:nnn {fontspec} {OpticalSize}
⟨*XE⟩
  {
    \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:nx {no-opticals} {\l_fontspec_fontname_tl} }
      }
  }
⟨/XE⟩
⟨*LU⟩
  {
    \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
  }
⟨/LU⟩
```

For other potentially font specific variation axes, there is a raw setter available:

```
\@@_keys_define_code:nnn {fontspec-opentype} {RawAxis}
  {
    \prop_gput_from_keyval:Nn \g_@@_rawvariations_prop {#1}
  }
```

1.4.2 Variation instances

```
\@@_keys_define_code:nnn {fontspec-opentype} {Instance}
  {
    \tl_gset:Nn \g_@@_instance_tl {#1}
  }
```

1.4.3 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}
  }
\keys_define:nn {fontspec}
  {
    FakeBold .code:n =
      {
        \@@_update_featstr:n {embolden=#1}
      },
    FakeBold .default:n = {1.5}
  }
```

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}
  }
```

1.4.4 Raw feature string

This allows savvy XƎTEX-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}
  }
```

File XIV

fontspec-code-feat-opentype.dtx

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\,=\,$N$ }
```

2 Regular key=val / tag definitions

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,
⟨XE⟩  mapping = tex-text
⟨LU⟩  +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.

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

2.2 Letters

```
\@@_define_opentype_feature_group:n {Letters}
\@@_define_opentype_feature:nnnnn   {Letters} {ResetAll} {} {}
  {
⟨LU⟩  +lower,-lower,+upper,-upper,+case,+cpsp,
    +smcp,+pcap,+c2sc,+c2pc,+unic,+rand,
    -smcp,-pcap,-c2sc,-c2pc,-unic,-rand
  }
```

```
⟨*LU⟩
```

```
\keys_define:nn {fontspec-opentype}
  {
    Letters / Uppercase .code:n = {
      \@@_make_OT_feature:nnn {} {+upper} {+lower}
      \@@_make_OT_feature:nnn {} {+case} {}
      \@@_make_OT_feature:nnn {} {+cpsp} {}
    },
  }
\@@_define_opentype_feature:nnnnn {Letters} {UppercaseOff} {} {-upper} {+case,+cpsp}
\@@_define_opentype_feature:nnnnn {Letters} {UppercaseReset} {} {} {+upper,-upper}
\@@_define_opentype_onoffreset:nnnnn {Letters} {Lowercase} {} {lower} {+upper,+case,+cpsp}
⟨/LU⟩
\@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {smcp} {+pcap,+unic}
\@@_define_opentype_onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic}
\@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+unic
\@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+uni
\@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {unic} {}
\@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {}
```

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.

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

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} {+s
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior}           {subs} {subs} {+s
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal}            {ordn} {ordn} {+s
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator}          {numr} {numr} {+s
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator}        {dnom} {dnom} {+s
\@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {+s
```

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} {}
```

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} {}
```

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}
```

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}
```

```
\@@_define_opentype_feature_group:n  {LocalForms}
\@@_define_opentype_feature:nnnnn    {LocalForms} {On}    {locl} {+locl} {}
\@@_define_opentype_feature:nnnnn    {LocalForms} {Off}   {locl} {-locl} {}
\@@_define_opentype_feature:nnnnn    {LocalForms} {Reset} {} {}  {+locl,-locl}
```

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} {Titling}          {titl} {titl} {}
\@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps}      {titl} {titl} {} % backwards c
\@@_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}
\@@_define_opentype_onoffreset:nnnnn {Style} {Uppercase} {case} {case} {}
```

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,+jp83
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified}  {smpl} {smpl} {+trad,+jp78,+jp83
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978}     {jp78} {jp78} {+trad,+smpl,+jp83
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983}     {jp83} {jp83} {+trad,+smpl,+jp78
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990}     {jp90} {jp90} {+trad,+smpl,+jp78
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004}     {jp04} {jp04} {+trad,+smpl,+jp78
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert}      {expt} {expt} {+trad,+smpl,+jp78
\@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC}         {nlck} {nlck} {+trad,+smpl,+jp78
```

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} {+
```

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,+
```

3 OpenType features that need numbering

3.1 Alternate

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

```
⟨*LU⟩
\keys_define:nn {fontspec-opentype}
  {
    Alternate / Random  .code:n =
      { \@@_make_OT_feature:nnn {salt}{ +salt = random }{} } ,
  }
⟨/LU⟩
```

```
\aliasfontfeature{Alternate}{StylisticAlternates}
```

3.2 Variant / StylisticSet

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

```
\aliasfontfeature{Variant}{StylisticSet}
```

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`.

3.4 Annotation

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

3.5 Ornament

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

4 Script and Language

4.1 Script

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

```
  {
    Script .choice: ,
    Script .groups:n = {opentype} ,
  }
\cs_new:Nn \fontspec_new_script:nn
  {
    \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
      {
⟨debug⟩\typeout{Trying~[Script=#1]}
        \bool_set_false:N \l_@@_scriptlang_exist_bool
        \clist_map_inline:nn {#2}
          {
            \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
              {
⟨debug⟩\typeout{Script~tag~found:~####1}
                \tl_set:Nn \l_@@_script_name_tl {#1}
                \tl_set:Nn \l_@@_script_tl {####1}
                \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
                \bool_set_true:N \l_@@_scriptlang_exist_bool
                \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
                \clist_map_break:
              }
          }
```

If not found give a warning but load it anyway:

```
        \bool_if:NF \l_@@_scriptlang_exist_bool
          {
⟨debug⟩\typeout{Script~not~found!}
            \@@_info:nxx {script-not-exist} {\l_fontspec_fontname_tl} {#1}
            \clist_set:Nn \l_tmpa_clist {#2}
            \clist_get:NN \l_tmpa_clist \l_@@_script_tl
            \exp_args:Noo \@@_check_script:NnF \l_@@_fontface_cs_tl \l_@@_script_tl
              {
                \tl_set:Nn \l_@@_script_name_tl {#1}
                \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
                \tl_gset:Nx \g_@@_single_feat_tl { script=\l_@@_script_tl }
              }
          }
      }
    }
  }
```

When script is not explicitly requested, use this list:

```
\clist_new:N \g_@@_default_scripts_clist
\cs_new:Nn \fontspec_default_script:n
  {
    \clist_gset:Nn \g_@@_default_scripts_clist {#1}
  }
\fontspec_default_script:n {latn,DFLT}
```

```
\keys_define:nn {fontspec-opentype} { Script / CustomDefault .code:n =
  {
⟨debug⟩\typeout{Trying~CustomDefault~Script}
```

```
  \bool_set_false:N \l_@@_scriptlang_exist_bool
  \clist_map_inline:Nn \g_@@_default_scripts_clist
   {
    \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {##1}
     {
⟨debug⟩\typeout{Script~tag~found:~##1}
      \tl_set:Nn \l_@@_script_name_tl {Default (##1)}
      \tl_set:Nn \l_@@_script_tl {##1}
      \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
      \bool_set_true:N \l_@@_scriptlang_exist_bool
      \tl_gset:Nx \g_@@_single_feat_tl { script=##1 }
      \clist_map_break:
     }
   }
  \bool_if:NF \l_@@_scriptlang_exist_bool
   {
⟨debug⟩\typeout{Script~not~found!}
    \tl_clear:N \l_@@_script_name_tl
   }
 }
}
```

4.2 Language

```
\keys_define:nn {fontspec-opentype}
 {
  Language .choice: ,
  Language .groups:n = {opentype} ,
 }
\cs_new:Nn \fontspec_new_lang:nn
 {
  \keys_define:nn {fontspec-opentype} { Language / #1 .code:n =
   {
    \bool_set_false:N \l_@@_scriptlang_exist_bool
    \clist_map_inline:nn {#2}
     {
      \exp_args:No \@@_check_lang:NnTF \l_@@_fontface_cs_tl {####1}
       {
        \tl_set:Nn \l_@@_lang_tl {####1}
        \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
        \tl_gset:Nx \g_@@_single_feat_tl { language=####1 }
        \bool_set_true:N \l_@@_scriptlang_exist_bool
        \clist_map_break:
       }
     }
```

If not found give a warning but load it anyway:

```
    \bool_if:NF \l_@@_scriptlang_exist_bool
     {
⟨debug⟩\typeout{Lang~not~found!}
      \@@_info:nx {language-not-exist} {#1}
      \clist_set:Nn \l_tmpa_clist {#2}
      \clist_get:NN \l_tmpa_clist \l_@@_lang_tl
```

```
            \exp_args:Noo \@@_check_lang:NnF \l_@@_fontface_cs_tl \l_@@_lang_tl
              {
                \tl_set:Nn \l_@@_lang_name_tl {#1}
                \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
                \tl_gset:Nx \g_@@_single_feat_tl { language=\l_@@_lang_tl }
              }
          }
      }
    }
  }
```

Language=Default These are special-cased to avoid the additional logic above. From memory, the OpenType default language is hardcoded to have a zero value, although this might be some XeTeX-specific thing.

```
\clist_new:N \g_@@_default_langs_clist
\cs_new:Nn \fontspec_default_lang:n
  {
    \clist_gset:Nn \g_@@_default_langs_clist {#1}
  }
\fontspec_default_lang:n {dflt}
\@@_keys_define_code:nnn {fontspec-opentype} { Language / Default }
  {
    \tl_set:Nn \l_@@_lang_tl {dflt}
    \int_zero:N \l_@@_language_int
    \tl_gset:Nn \g_@@_single_feat_tl { language=dflt }
  }
```

5 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}
```

File XV

fontspec-code-scripts.dtx

1 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{Chorasmian}{chrs}
\newfontscript{CJK~Ideographic}{hani}
\newfontscript{Coptic}{copt}
\newfontscript{Cypriot~Syllabary}{cprt}
\newfontscript{Cypro-Minoan}{cpmn}
\newfontscript{Cyrillic}{cyrl}
\newfontscript{Default}{DFLT}
\newfontscript{Deseret}{dsrt}
\newfontscript{Devanagari}{dev2,deva}
\newfontscript{Dives~Akuru}{diak}
\newfontscript{Dogra}{dogr}
\newfontscript{Duployan}{dupl}
\newfontscript{Egyptian~Hieroglyphs}{egyp}
\newfontscript{Elbasan}{elba}
\newfontscript{Elymaic}{elym}
\newfontscript{Ethiopic}{ethi}
\newfontscript{Garay}{gara}
\newfontscript{Georgian}{geor}
\newfontscript{Glagolitic}{glag}
\newfontscript{Gothic}{goth}
```

```
\newfontscript{Grantha}{gran}
\newfontscript{Greek}{grek}
\newfontscript{Gujarati}{gjr2,gujr}
\newfontscript{Gunjala~Gondi}{gong}
\newfontscript{Gurmukhi}{gur2,guru}
\newfontscript{Gurung~Khema}{gukh}
\newfontscript{Hangul~Jamo}{jamo}
\newfontscript{Hangul}{hang}
\newfontscript{Hanifi~Rohingya}{rohg}
\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{Kawi}{kawi}
\newfontscript{Kayah~Li}{kali}
\newfontscript{Kharosthi}{khar}
\newfontscript{Khitan~Small~Script}{kits}
\newfontscript{Khmer}{khmr}
\newfontscript{Khojki}{khoj}
\newfontscript{Khudawadi}{sind}
\newfontscript{Kirat~Rai}{krai}
\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{Makasar}{maka}
\newfontscript{Malayalam}{mlm2,mlym}
\newfontscript{Mandaic}{mand}
\newfontscript{Manichaean}{mani}
\newfontscript{Marchen}{marc}
\newfontscript{Masaram~Gondi}{gonm}
\newfontscript{Math}{math}
\newfontscript{Medefaidrin}{medf}
\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{Nag~Mundari}{nagm}
\newfontscript{Nandinagari}{nand}
\newfontscript{Newa}{newa}
\newfontscript{Nushu}{nshu}
\newfontscript{Nyiakeng~Puachue~Hmong}{hmnp}
\newfontscript{Odia}{ory2,orya}
\newfontscript{Ogham}{ogam}
\newfontscript{Ol~Chiki}{olck}
\newfontscript{Ol~Onal}{onao}
\newfontscript{Old~Italic}{ital}
\newfontscript{Old~Hungarian}{hung}
\newfontscript{Old~North~Arabian}{narb}
\newfontscript{Old~Permic}{perm}
\newfontscript{Old~Persian~Cuneiform}{xpeo}
\newfontscript{Old~Sogdian}{sogo}
\newfontscript{Old~South~Arabian}{sarb}
\newfontscript{Old~Turkic}{orkh}
\newfontscript{Old~Uyghur}{ougr}
\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{Sogdian}{sogd}
\newfontscript{Sora~Sompeng}{sora}
\newfontscript{Sumero-Akkadian~Cuneiform}{xsux}
\newfontscript{Sundanese}{sund}
\newfontscript{Sunuwar}{sunu}
\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{Tangsa}{tnsa}
\newfontscript{Tangut}{tang}
\newfontscript{Telugu}{tel2,telu}
\newfontscript{Thaana}{thaa}
\newfontscript{Thai}{thai}
\newfontscript{Tibetan}{tibt}
\newfontscript{Tifinagh}{tfng}
\newfontscript{Tirhuta}{tirh}
\newfontscript{Todhri}{todr}
\newfontscript{Toto}{toto}
\newfontscript{Tulu-Tigalari}{tutg}
\newfontscript{Ugaritic~Cuneiform}{ugar}
\newfontscript{Vai}{vai~}
\newfontscript{Vithkuqi}{vith}
\newfontscript{Wancho}{wcho}
\newfontscript{Warang~Citi}{wara}
\newfontscript{Yezidi}{yezi}
\newfontscript{Yi}{yi~~}
\newfontscript{Zanabazar~Square}{zanb}
```

For convenience or backwards compatibility:

```
\newfontscript{CJK}{hani}
\newfontscript{Kana}{kana}
\newfontscript{Maths}{math}
\newfontscript{N'ko}{nko~}
\newfontscript{Oriya}{ory2,orya}
```

File XVI

fontspec-code-lang.dtx

1 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{Turkish}{TRK,TUR}
\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}
```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for XƎTEX.

1.1 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 }
   }
 }
```

1.2 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}
```

1.3 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}
```

1.4 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}
```

1.5 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}
```

1.6 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}
```

1.7 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}
```

1.8 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}
          }
      }
  }
```

1.9 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}
      }
  }
```

1.10 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} {Titling} {19} {4}
\@@_define_aat_feature:nnnn      {Style} {TitlingCaps} {19} {4} % backwards compat
\@@_define_aat_feature:nnnn      {Style} {TallCaps} {19} {5}
```

1.11 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}
```

1.12 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}
```

1.13 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}
```

File XVIII

fontspec-code-enc.dtx

1 Extended font encodings

`\EncodingCommand`

```
\DeclareDocumentCommand \EncodingCommand { m O{} O{} m }
  {
    \bool_if:NF \l_@@_defining_encoding_bool
      { \@@_error:nn {only-inside-encdef} \EncodingCommand }
    \DeclareTextCommand{#1}{\UnicodeEncodingName}[#2][#3]{#4}
  }
```

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

`\DeclareUnicodeEncoding`

```
\DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
  {
    \DeclareFontEncoding{#1}{}{}
    \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 of 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 of 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 of definition for* `\UndeclareComposite`. *This function is documented on page* **??**.)

File XIX

fontspec-code-math.dtx

1 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_gset_true:N  \g_@@_pkg_euler_loaded_bool }
  { \bool_gset_false:N \g_@@_pkg_euler_loaded_bool }
```

```
\cs_new:Nn \fontspec_setup_maths:
 {
  \@ifpackageloaded{euler}
   {
    \bool_if:NTF \g_@@_pkg_euler_loaded_bool
     { \bool_gset_true:N \g_@@_math_euler_bool }
     { \@@_error:n {euler-too-late} }
   }
   {}
  \@ifpackageloaded{lucbmath}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
  \@ifpackageloaded{lucidabr}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
  \@ifpackageloaded{lucimatx}{ \bool_gset_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="603A \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}
    \renewcommand{\hbar}{{\mathchar"AF\mkern-9mu h}}% TODO: test with other fonts
  }
}
```

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\shapedefault
\SetSymbolFont{operators}{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\shapedefau
\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\shapedefault
\SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g_@@_mathsf_tl\mddefault\shapedefault
\SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g_@@_mathtt_tl\mddefault\shapedefault
\SetSymbolFont{operators}{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\shapedefault
\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\shapedefau
  \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\bfdefault\shapedefau
```

```
    \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\itdefault
   }
  \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\shapedefault
  \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\shapedefault
 }
```

(*End of 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_gset_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{arevmath}       {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{eulervm}        {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathdesign}     {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{concmath}       {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{cmbright}       {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathesf}        {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{gfsartemisia}   {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{gfsneohellenic} {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{iwona}
   {
    \ifx\define@iwona@mathversions a\bool_set_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{kpfonts}{\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{kmath}  {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{kurier}
   {
    \ifx\define@kurier@mathversions a\bool_set_false:N \g_@@_math_bool\fi
   }{}
  \@ifpackageloaded{fouriernc}    {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{fourier}      {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{lmodern}      {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathpazo}     {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mathptmx}     {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{MinionPro}    {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{unicode-math} {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{breqn}        {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{pxfonts}      {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{txfonts}      {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{newpxmath}    {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{newtxmath}    {\bool_gset_false:N \g_@@_math_bool}{}
  \@ifpackageloaded{mtpro2}       {\bool_gset_false:N \g_@@_math_bool}{}
```

```
   \bool_if:NT \g_@@_math_bool
    {
     \@@_info:n {setup-math}
     \fontspec_setup_maths:
    }
  }
\AtBeginDocument{\fontspec_maybe_setup_maths:}
```

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

File XX

fontspec-code-closing.dtx

1 Closing code

1.1 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.} }
  }
```

File XXI

fontspec-code-xfss.dtx

1 Changes/additions to the NFSS

`\strong`
`\strongenv`

`\strong {⟨text⟩}`
`\begin{strongenv} ⟨text⟩ \end{strongenv}`

Typesets text in the 'strong' font. NFSS series equivalent to `\emph`. Can be nested.

`\strongfontdeclare`

`\strongfontdeclare {⟨comma-separated font switch declarations⟩}`

Define the behaviour of nested `\strong` commands.

`\strongreset`

`\renewcommand \strongreset {⟨font switch declarations⟩}`

Define the behaviour when a `\strong` command is nested deeper than the definitions provided by `\strongfontdeclare`. By default this is ⟨*empty*⟩ — i.e., bold on top of bold remains bold. In certain circumstances it may be appropriate to reset to a default state.

2 Implementation

```
⟨*fontspec⟩
```

2.1 Italic small caps and so on

```
\providecommand*\scitdefault{\scdefault\itdefault}
\providecommand*\scsldefault{\scdefault\sldefault}
\providecommand*\scswdefault{\scdefault\swdefault}
```

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 }
\cs_new:Nn \@@_merge_default_shapes:
  {
    \tl_const:cn { \@@_shape_merge:nn \shapedefault\scdefault    } {\scdefault}
    \tl_const:cn { \@@_shape_merge:nn \itdefault   \scdefault    } {\scitdefault}
    \tl_const:cn { \@@_shape_merge:nn \sldefault   \scdefault    } {\scsldefault}
    \tl_const:cn { \@@_shape_merge:nn \swdefault   \scdefault    } {\scswdefault}
    \tl_const:cn { \@@_shape_merge:nn \scdefault   \itdefault    } {\scitdefault}
    \tl_const:cn { \@@_shape_merge:nn \scdefault   \sldefault    } {\scsldefault}
    \tl_const:cn { \@@_shape_merge:nn \scdefault   \swdefault    } {\scswdefault}
    \tl_const:cn { \@@_shape_merge:nn \scsldefault \itdefault    } {\scitdefault}
    \tl_const:cn { \@@_shape_merge:nn \scitdefault \sldefault    } {\scsldefault}
    \tl_const:cn { \@@_shape_merge:nn \scitdefault \shapedefault } {\scdefault}
    \tl_const:cn { \@@_shape_merge:nn \scsldefault \shapedefault } {\scdefault}
  }
\@@_merge_default_shapes:
```

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:
  }
```

```
\cs_set_eq:NN \emfontdeclare \DeclareEmphSequence
```

2.2 Strong emphasis

`\strongfontdeclare`

```
\cs_set_protected:Npn \strongfontdeclare #1
  {
    \prop_gclear:N   \g_@@_strong_prop
    \int_zero:N      \l_@@_strongdef_int

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

(End of definition for `\strongfontdeclare`*. This function is documented on page 133.)*

`\strongenv`

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

⟨debug⟩ \typeout{Strong~ level:~\int_use:N \l_@@_strong_int}
    \prop_get:NeNT \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:NeNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_swit
      { \l_@@_strong_switch_tl }
      {
        \int_zero:N \l_@@_strong_int
        \strongreset
      }

  }
```

(*End of definition for* `\strongenv`. *This function is documented on page 133.*)

`\strong`

```
\DeclareTextFontCommand{\strong}{\strongenv}
```

(*End of definition for* `\strong`. *This function is documented on page 133.*)

`\strongreset`

```
\cs_set:Npn \strongreset {}
```

(*End of definition for* `\strongreset`. *This function is documented on page 133.*)

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

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

(*End of definition for* `\reset@font`.)

2.3 Defaults

```
\strongfontdeclare{\bfseries}
⟨/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.