

The fontspec package

Font selection for X_YLaTeX and LuaLaTeX

WILL ROBERTSON

With contributions by Khaled Hosny,
Philipp Gesang, Joseph Wright, and others.

<http://wsprr.io/fontspec/>

2020/02/21 v2.7i

Contents

I fontspec.dtx	5
1 Package declaration	5
1.1 Lua header	6
II fontspec-code-load.dtx	7
1 The fontspec.sty loading file	7
III fontspec-code-vars.dtx	8
1 Declaration of variables	8
IV fontspec-code-msg.dtx	13
1 Error/warning/info messages	13
1.1 Errors	13
1.2 Warnings	14
1.3 Info messages	16
V fontspec-code-opening.dtx	18
1 Opening code	18
1.1 Package options	18
1.2 Encodings	18

1.3	Generic functions	19
1.4	expl3 variants	20
VI	fontspec-code-fontload.dtx	21
1	expl3 interface for primitive font loading	21
VII	fontspec-code-interfaces.dtx	23
1	User commands	23
VIII	fontspec-code-user.dtx	27
1	User command internals	27
1.1	Font selection	27
1.2	Font feature selection	30
1.3	Defining new font features	32
1.4	High level conditionals	34
1.5	\oldstylenums and \liningnums	35
IX	fontspec-code-api.dtx	36
1	Programmer's interface	36
X	fontspec-code-internal.dtx	42
1	Internals	42
1.1	The main function for setting fonts	42
1.2	Setting font shapes in a family	49
1.3	Initialisation	59
1.4	Miscellaneous	60
XI	fontspec-code-opentype.dtx	62
1	OpenType definitions code	62
1.1	Adding features when loading fonts	63
1.2	OpenType feature information	67
XII	fontspec-code-graphite.dtx	70
1	Graphite/AAT code	70

XIII	fontspec-code-keyval.dtx	72
1	Font loading (keyval) definitions	72
1.1	Pre-pre-parsing stages	72
1.2	Pre-parsed features	74
1.3	Font faces	75
1.4	General font-independent features	78
XIV	fontspec-code-feat-opentype.dtx	87
1	OpenType feature definitions	87
2	Regular key=val / tag definitions	87
2.1	Ligatures	87
2.2	Letters	87
2.3	Numbers	88
2.4	Vertical position	88
2.5	Contextuals	88
2.6	Diacritics	89
2.7	Kerning	89
2.8	Fractions	89
2.9	Style	90
2.10	CJK shape	90
2.11	Character width	91
2.12	Vertical	91
3	OpenType features that need numbering	91
3.1	Alternate	91
3.2	Variant / StylisticSet	92
3.3	CharacterVariant	92
3.4	Annotation	93
3.5	Ornament	93
4	Script and Language	93
4.1	Script	93
4.2	Language	94
5	Backwards compatibility	95
XV	fontspec-code-scripts.dtx	96
1	Font script definitions	96
XVI	fontspec-code-lang.dtx	100
1	Font language definitions	100

XVII	fontspec-code-feat-aat.dtx	108
1	AAT feature definitions	108
1.1	Ligatures	108
1.2	Letters	108
1.3	Numbers	109
1.4	Contextuals	109
1.5	Diacritics	109
1.6	Vertical position	109
1.7	Fractions	109
1.8	Alternate	109
1.9	Variant / StylisticSet	110
1.10	Style	110
1.11	CJK shape	110
1.12	Character width	111
1.13	Annotation	111
XVIII	fontspec-code-enc.dtx	112
1	Extended font encodings	112
XIX	fontspec-code-math.dtx	115
1	Selecting maths fonts	115
XX	fontspec-code-closing.dtx	120
1	Closing code	120
1.1	Finishing up	120
XXI	fontspec-code-xfss.dtx	121
1	Changes to the NFSS	121
1.1	Italic small caps and so on	121
1.2	Emphasis	122
1.3	Strong emphasis	124
Index		126

File I

fontspec.dtx

1 Package declaration

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

```
1 <*dtx>
2 \gdef\FONTSPECDTX{
3   \DTX{fontspec.dtx}
4   \DTX{fontspec-code-load.dtx}
5   \DTX{fontspec-code-vars.dtx}
6   \DTX{fontspec-code-msg.dtx}
7   \DTX{fontspec-code-opening.dtx}
8   \DTX{fontspec-code-fontload.dtx}
9   \DTX{fontspec-code-interfaces.dtx}
10  \DTX{fontspec-code-user.dtx}
11  \DTX{fontspec-code-api.dtx}
12  \DTX{fontspec-code-internal.dtx}
13  \DTX{fontspec-code-opentype.dtx}
14  \DTX{fontspec-code-graphite.dtx}
15  \DTX{fontspec-code-keyval.dtx}
16  \DTX{fontspec-code-feat-opentype.dtx}
17  \DTX{fontspec-code-scripts.dtx}
18  \DTX{fontspec-code-lang.dtx}
19  \DTX{fontspec-code-feat-aat.dtx}
20  \DTX{fontspec-code-enc.dtx}
21  \DTX{fontspec-code-math.dtx}
22  \DTX{fontspec-code-closing.dtx}
23  \DTX{fontspec-code-xfss.dtx}
24 }
25 </dtx>
```

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

```
26 <*dtx>
27 \def\tmpa{plain}
28 \ifx\tmpa\fmtname\expandafter\endinput\fi
29 </dtx>
```

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

```
30 <*dtx>
31 \title{
32   The \textsf{fontspec} package\\
33   Font selection for \XeLaTeX\ and \LuaLaTeX
34 }
35 \author{
36   \textsc{Will Robertson}\\
37   With contributions by Khaled Hosny,\\
38   Philipp Gesang, Joseph Wright, and others.\\
39   \url{http://wspr.io/fontspec/}
```

```

40 }
41 </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.

```

42 <fontspec>\RequirePackage{xparse}
43 <fontspec & load>\ProvidesExplPackage{fontspec}%
44 <fontspec & XE>\ProvidesExplPackage{fontspec-xetex}%
45 <fontspec & LU>\ProvidesExplPackage{fontspec-luatex}%
46 <*dtx>
47 \RequirePackage{xparse}
48 \ProvidesExplFile{fontspec.dtx}
49 </dtx>
50 <*fontspec>
51 {2020/02/21}{2.7i}{Font selection for XeLaTeX and LuaLaTeX}
52 </fontspec>

```

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

```

53 <*dtx>
54 \GetFileInfo{fontspec.dtx}
55 \date{\filedate \quad \fileversion}
56 </dtx>

```

1.1 Lua header

```

57 <lua>fontspec = fontspec or {}
58 <lua>local fontspec = fontspec
59 <lua>fontspec.module = {
60 <lua>  name = "fontspec",
61 <lua>  version = "2.7i",
62 <lua>  date = "2020/02/21",
63 <lua>  description = "Font selection for XeLaTeX and LuaLaTeX",
64 <lua>  author = "Khaled Hosny, Philipp Gesang, Will Robertson",
65 <lua>  copyright = "Khaled Hosny, Philipp Gesang, Will Robertson",
66 <lua>  license = "LPPL v1.3c"
67 <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'.

```
1 <@@=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.

```
2 <*load>
```

Lua^AT_EX

```
3 \sys_if_engine_luatex:T
4 {
5   \RequirePackage{luaotfload}
6   \lua_now:e{require("fontspec")}
7   \RequirePackageWithOptions{fontspec-luatex}
8   \endinput
9 }
```

X_EL^AT_EX

```
10 \sys_if_engine_xetex:T
11 {
12   \RequirePackageWithOptions{fontspec-xetex}
13   \endinput
14 }
```

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

```
15 \msg_new:nnn {fontspec} {cannot-use-pdftex}
16 {
17   The~ fontspec~ package~ requires~ either~ XeTeX~ or~ LuaTeX.\\\
18   You~ must~ change~ your~ typesetting~ engine~ to,~ e.g.,~
19   "xelatex"~ or~ "lualatex"~ instead~ of~ "latex"~ or~ "pdflatex".
20 }
21 \msg_fatal:nn {fontspec} {cannot-use-pdftex}
```

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

```
22 \endinput
23 </load>
```

File III

fontspec-code-vars.dtx

1 Declaration of variables

This file consists solely of declaration of variables used by fontspec. In some cases these variables are also initialised with default values. In time I would like to move these initialisations

Booleans

`\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.

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

(End definition for \l_@@_firsttime_bool. This function is documented on page ??.)

```
2 \bool_new:N \l_@@_nobf_bool
```

```
3 \bool_new:N \l_@@_noit_bool
```

```
4 \bool_new:N \l_@@_nosc_bool
```

```
5 \bool_new:N \l_@@_check_bool
```

```
6 \bool_new:N \l_@@_tfm_bool
```

```
7 \bool_new:N \l_@@_atsui_bool
```

```
8 \bool_new:N \l_@@_ot_bool
```

```
9 \bool_new:N \l_@@_mm_bool
```

```
10 \bool_new:N \l_@@_harfbuzz_bool
```

```
11 \bool_new:N \l_@@_graphite_bool
```

```
12 \bool_new:N \l_@@_fontcfg_bool
```

```
13 \bool_set_true:N \l_@@_fontcfg_bool
```

For dealing with legacy maths:

```
14 \bool_new:N \g_@@_math_euler_bool
```

```
15 \bool_new:N \g_@@_math_lucida_bool
```

```
16 \bool_new:N \g_@@_pkg_euler_loaded_bool
```

For package options:

```
17 \bool_new:N \g_@@_cfg_bool
```

```
18 \bool_new:N \g_@@_math_bool
```

```
19 \bool_new:N \g_@@_euenc_bool
```

```
20 \bool_new:N \l_@@_tmpa_bool
```

```
21 \bool_new:N \l_@@_disable_defaults_bool
```

```
22 \bool_new:N \l_@@_alias_bool
```

```
23 \bool_new:N \l_@@_external_bool
```

```
24 \bool_new:N \l_@@_defining_encoding_bool
```

```
25 \bool_new:N \l_@@_scriptlang_exist_bool
```

```
26 \bool_new:N \g_@@_em_normalise_slant_bool
```

```
27 \bool_new:N \l_@@_proceed_bool
```


`\l_@@_never_check_bool` It is 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!

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

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

Counters

```
29 \int_new:N \l_@@_script_int
30 \int_new:N \l_@@_language_int
31 \int_new:N \l_@@_strnum_int
32 \int_new:N \l_@@_tmp_int
33 \int_new:N \l_@@_tmpa_int
34 \int_new:N \l_@@_tmpb_int
35 \int_new:N \l_@@_tmpc_int
36 \int_new:N \l_@@_em_int
37 \int_new:N \l_@@_emdef_int
38 \int_new:N \l_@@_strong_int
39 \int_new:N \l_@@_strongdef_int
```

Floats

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

Dimensions

```
42 \dim_new:N \l_@@_tmpa_dim
43 \dim_new:N \l_@@_tmpb_dim
44 \dim_new:N \l_@@_tmpc_dim
```

Sequences

```
45 \seq_new:N \l_@@_bf_series_seq
```

Comma-lists

```
46 \clist_new:N \g_@@_default_fontopts_clist
47 \clist_new:N \g_@@_all_keyval_modules_clist
48 \clist_new:N \l_@@_sizefeat_clist
49 \clist_set:Nn \l_@@_sizefeat_clist {Size={-}}
50 \clist_new:N \l_@@_extensions_clist
51 \clist_new:N \l_@@_fontopts_clist
52 \clist_new:N \l_@@_family_fontopts_clist
53 \clist_new:N \l_@@_all_features_clist
54 \clist_new:N \l_@@_leftover_clist
55 \clist_new:N \l_@@_keys_leftover_clist
56 \clist_new:N \l_@@_sizing_leftover_clist
57 \clist_new:N \l_@@_fontfeat_clist
58 \clist_new:N \l_@@_fontfeat_curr_clist
59 \clist_new:N \l_@@_arg_clist
60 \clist_new:N \l_@@_this_feat_clist
```

```

61 \clist_new:N \l_@@_fontfeat_up_clist
62 \clist_new:N \l_@@_fontfeat_bf_clist
63 \clist_new:N \l_@@_fontfeat_it_clist
64 \clist_new:N \l_@@_fontfeat_bfit_clist
65 \clist_new:N \l_@@_fontfeat_sl_clist
66 \clist_new:N \l_@@_fontfeat_bfsl_clist
67 \clist_new:N \l_@@_fontfeat_sc_clist

```

Property lists

```

68 \prop_new:N \g_@@_fontopts_prop
69 \prop_new:N \l_@@_nfss_prop
70 \prop_new:N \l_@@_nfssfont_prop
71 \prop_new:N \g_@@_OT_features_prop
72 \prop_new:N \g_@@_all_opentype_feature_names_prop
73 \prop_new:N \g_@@_em_prop
74 \prop_new:N \g_@@_strong_prop
75 \prop_new:N \g_@@_fontid_family_prop
76 \prop_new:N \g_@@_family_int_prop

```

Token lists

Visible (perhaps?)

```

77 \tl_new:N \l_fontspeg_family_tl
78 \tl_new:N \g_fontspeg_encoding_tl
79 \tl_new:N \l_fontspeg_fontname_tl

```

2e interactions

```

80 \tl_clear_new:N \UTFencname
81 \tl_clear_new:N \cyrillicencoding
82 \tl_clear_new:N \latinencoding

```

Renderer/shaper

```

83 \tl_new:N \l_@@_renderer_tl
84 \tl_new:N \l_@@_mode_tl
85 \tl_new:N \l_@@_shaper_tl

86 \tl_new:N \g_@@_defined_shapes_tl
87 \tl_new:N \g_@@_single_feat_tl
88 \tl_new:N \l_@@_basename_tl
89 \tl_new:N \g_@@_curr_series_tl
90 \tl_new:N \l_@@_curr_fontname_tl
91 \tl_new:N \l_@@_curr_bfname_tl
92 \tl_new:N \l_@@_ext_filename_tl
93 \tl_new:N \l_@@_extension_tl
94 \tl_new:N \l_@@_font_path_tl
95 \tl_new:N \l_@@_fontid_tl
96 \tl_new:N \l_@@_fontname_tl
97 \tl_new:N \l_@@_options_tl
98 \tl_new:N \l_@@_saved_fontname_tl
99 \tl_new:N \l_@@_prev_unicode_name_tl

```

```

100 \tl_new:N \g_@@_nfss_enc_tl
101 \tl_new:N \g_@@_nfss_family_tl
102 \tl_new:N \l_@@_nfss_sc_tl
103 \tl_new:N \l_@@_nfss_tl
104 \tl_new:N \l_@@_nfss_fam_tl

105 \tl_new:N \l_@@_size_tl
106 \tl_new:N \l_@@_sizedfont_tl
107 \tl_new:N \l_@@_this_font_tl
108 \tl_new:N \l_@@_ttc_index_tl
109 \tl_new:N \l_@@_smcp_shape_tl

```

EM and STRONG

```

110 \tl_new:N \l_@@_emshape_query_tl
111 \tl_new:N \l_@@_em_switch_tl
112 \tl_new:N \l_@@_strong_switch_tl

```

Scratch variables

```

113 \tl_new:N \l_@@_tmp_tl
114 \tl_new:N \l_@@_tmpa_tl
115 \tl_new:N \l_@@_tmpb_tl
116 \tl_new:N \l_@@_em_tmp_tl
117 \tl_new:N \l_@@_strong_tmp_tl

```

Maths fonts

```

118 \tl_new:N \g_@@_mathrm_tl
119 \tl_new:N \g_@@_bfmathrm_tl
120 \tl_new:N \g_@@_mathsf_tl
121 \tl_new:N \g_@@_mathtt_tl

```

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

```

122 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
123 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
124 \tl_gset:Nn \g_@@_mathtt_tl {\ttdefault}

125 \tl_new:N \l_@@_family_label_tl
126 \tl_new:N \l_@@_fake_slant_tl
127 \tl_new:N \l_@@_fake_embolden_tl

```

Internal font names

```

128 \tl_new:N \l_@@_fontname_up_tl
129 \tl_new:N \l_@@_fontname_bf_tl
130 \tl_new:N \l_@@_fontname_it_tl
131 \tl_new:N \l_@@_fontname_bfit_tl
132 \tl_new:N \l_@@_fontname_sl_tl
133 \tl_new:N \l_@@_fontname_bfsl_tl
134 \tl_new:N \l_@@_fontname_sc_tl

```

Script and Language

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

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

Generic font features

```
141 \tl_new:N \l_@@_scale_tl
142 \tl_new:N \l_@@_hyphenchar_tl
143 \tl_new:N \l_@@_hexcol_tl
144 \tl_new:N \l_@@_opacity_tl
145 \tl_new:N \l_@@_optical_size_tl
146 \tl_new:N \l_@@_mapping_tl
147 \tl_new:N \l_@@_punctspace_adjust_tl
148 \tl_new:N \l_@@_wordspace_adjust_tl
149 \tl_new:N \l_@@_postadjust_tl

150 \tl_const:Nn \c_@@_hexcol_tl {000000}
151 \tl_const:Nn \c_@@_opacity_tl {FF~}
152 \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.

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

Font families

```
155 \tl_new:N \l_@@_rmfamily_family_tl
156 \tl_new:N \l_@@_sffamily_family_tl
157 \tl_new:N \l_@@_ttfamily_family_tl
158 \tl_new:N \l_@@_rmfamily_encoding_tl
159 \tl_new:N \l_@@_sffamily_encoding_tl
160 \tl_new:N \l_@@_ttfamily_encoding_tl
```

File IV

fontspec-code-msg.dtx

1 Error/warning/info messages

Shorthands for messages:

```
1 \cs_new:Npn \@@_error:n { \msg_error:nn {fontspec} }
2 \cs_new:Npn \@@_error:nn { \msg_error:nnn {fontspec} }
3 \cs_new:Npn \@@_error:nx { \msg_error:nxx {fontspec} }
4 \cs_new:Npn \@@_warning:n { \msg_warning:nn {fontspec} }
5 \cs_new:Npn \@@_warning:nx { \msg_warning:nxx {fontspec} }
6 \cs_new:Npn \@@_warning:nxx { \msg_warning:nxxx {fontspec} }
7 \cs_new:Npn \@@_info:n { \msg_info:nn {fontspec} }
8 \cs_new:Npn \@@_info:nx { \msg_info:nxx {fontspec} }
9 \cs_new:Npn \@@_info:nxx { \msg_info:nxxx {fontspec} }
10 \cs_new:Npn \@@_trace:n { \msg_trace:nn {fontspec} }
```

Allow messages to be written with spaces acting as normal:

```
11 \cs_generate_variant:Nn \msg_new:nnn {nxx}
12 \cs_generate_variant:Nn \msg_new:nnnn {nxxx}
13 \cs_new:Nn \@@_msg_new:nnn
14 { \msg_new:nxx {#1} {#2} { \tl_trim_spaces:n {#3} } }
15 \cs_new:Nn \@@_msg_new:nnnn
16 { \msg_new:nxxx {#1} {#2} { \tl_trim_spaces:n {#3} } { \tl_trim_spaces:n {#4} } }
17 \char_set_catcode_space:n {32}
```

1.1 Errors

```
18 \@@_msg_new:nnn {fontspec} {only-inside-encdef}
19 {
20   \exp_not:N#1can only be used in the second argument
21   to \string\DeclareUnicodeEncoding.
22 }
23 \@@_msg_new:nnn {fontspec} {no-size-info}
24 {
25   Size information must be supplied.\\
26   For example, SizeFeatures={Size={8-12},...}.
27 }
28 \@@_msg_new:nnnn {fontspec} {font-not-found}
29 {
30   The font "#1" cannot be found.
31 }
32 {
33   A font might not be found for many reasons.\\
34   Check the spelling, where the font is installed etc. etc.\\
35   When in doubt, ask someone for help!
36 }
37 \@@_msg_new:nnnn {fontspec} {rename-feature-not-exist}
38 {
```

```

39   The feature #1 doesn't appear to be defined.
40   }
41   {
42   It looks like you're trying to rename a feature that doesn't exist.
43   }
44   \@@_msg_new:nnn {fontspec} {no-glyph}
45   {
46   '#1' does not contain glyph #2.
47   }
48   \@@_msg_new:nxxx {fontspec} {euler-too-late}
49   {
50   The euler package must be loaded BEFORE fontspec.
51   }
52   {
53   fontspec only overwrites euler's attempt to
54   define the maths text fonts if fontspec is
55   loaded after euler. Type <return> to proceed
56   with incorrect \string\mathit, \string\mathbf, etc.
57   }
58   \@@_msg_new:nxxx {fontspec} {no-xcolor}
59   {
60   Cannot load named colours without the xcolor package.
61   }
62   {
63   Sorry, I can't do anything to help. Instead of loading
64   the color package, use xcolor instead.
65   }
66   \@@_msg_new:nxxx {fontspec} {unknown-color-model}
67   {
68   Error loading colour `#1'; unknown colour model.
69   }
70   {
71   Sorry, I can't do anything to help. Please report this error
72   to my developer with a minimal example that causes the problem.
73   }
74   \@@_msg_new:nxxx {fontspec} {not-in-addfontfeatures}
75   {
76   The "#1" font feature cannot be used in \string\addfontfeatures.
77   }
78   {
79   This is due to how TeX loads fonts; such settings
80   are global so adding them mid-document within a group causes
81   confusion. You'll need to define multiple font families to achieve
82   what you want.
83   }

```

1.2 Warnings

```

84   \@@_msg_new:nnn {fontspec} {tu-clash}
85   {
86   I have found the tuenc.def encoding definition file but the TU encoding is not
87   defined by the LaTeX2e kernel; attempting to correct but you really should update

```

```

88   to the latest version of LaTeX2e.
89   }
90   \@@_msg_new:nnn {fontspec} {tu-missing}
91   {
92     The TU encoding seems to be missing; please update to the latest version of LaTeX2e.
93   }
94   \@@_msg_new:nnn {fontspec} {addfontfeatures-ignored}
95   {
96     \string\addfontfeature (s) ignored \msg_line_context;
97     it cannot be used with a font that wasn't selected by a fontspec command.\\
98     \\
99     The current font is "\use:c{font@name}".\\
100    \int_compare:nTF { \clist_count:n {#1} = 1 }
101      { The requested feature is "#1". }
102      { The requested features are "#1". }
103    }
104    \@@_msg_new:nnn {fontspec} {feature-option-overwrite}
105    {
106      Option '#2' of font feature '#1' overwritten.
107    }
108    \@@_msg_new:nnn {fontspec} {ot-tag-too-long}
109    {
110      OpenType tag '#1' is too long; script, language, and feature tags must be four characters or less.
111    }
112    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist}
113    {
114      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
115      for AAT font '\l_fontspec_fontname_tl'.
116    }
117    \@@_msg_new:nnn {fontspec} {aat-feature-not-exist-in-font}
118    {
119      AAT feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
120      in font '\l_fontspec_fontname_tl'.
121    }
122    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist}
123    {
124      '\l_keys_key_tl=\l_keys_value_tl' feature not supported
125      for OpenType font '\l_fontspec_fontname_tl'
126    }
127    \@@_msg_new:nnn {fontspec} {icu-feature-not-exist-in-font}
128    {
129      OpenType feature '\l_keys_key_tl=\l_keys_value_tl' (#1) not available
130      for font '\l_fontspec_fontname_tl'
131      with script '\l_@@_script_name_tl' and language '\l_@@_lang_name_tl'.
132    }
133    \@@_msg_new:nnn {fontspec} {no-opticals}
134    {
135      '#1' doesn't appear to have an Optical Size axis.
136    }
137    \@@_msg_new:nnn {fontspec} {language-not-exist}
138    {

```

```

139 Language '#1' not available
140 for font '\l_fontspec_fontname_tl'
141 with script '\l_@@_script_name_tl'.
142 }
143 \@@_msg_new:nnn {fontspec} {only-xetex-feature}
144 {
145   Ignored XeTeX-only feature: '#1'.
146 }
147 \@@_msg_new:nnn {fontspec} {only-luatex-feature}
148 {
149   Ignored LuaTeX-only feature: '#1'.
150 }
151 \@@_msg_new:nnn {fontspec} {unknown-renderer}
152 {
153   Renderer '#1' unknown. Assuming Harfbuzz with 'shaper=#1'.
154   Please raise a fontspec issue to add this shaper to the interface.
155 }
156 \@@_msg_new:nnn {fontspec} {no-mapping}
157 {
158   Input mapping not supported in LuaTeX.
159 }
160 \@@_msg_new:nnn {fontspec} {no-mapping-ligtext}
161 {
162   Input mapping not supported in LuaTeX.\\
163   Use "Ligatures=TeX" instead of "Mapping=tex-text".
164 }
165 \@@_msg_new:nnn {fontspec} {cm-default-obsolete}
166 {
167   The "cm-default" package option is obsolete.
168 }
169 \@@_msg_new:nnn {fontspec} {font-index-needs-ttc}
170 {
171   The "FontIndex" feature is only supported by TTC (TrueType Collection) fonts.\\
172   Feature ignored.
173 }
174 \@@_msg_new:nnn {fontspec} {feat-cannot-remove}
175 {
176   The "#1" feature cannot be deactivated. Request ignored.
177 }

```

1.3 Info messages

```

178 \@@_msg_new:nnn {fontspec} {defining-font}
179 {
180   Font family '\g_@@_nfss_family_tl' created for font '#2'
181   with options [\l_@@_all_features_clist].\\
182   \\
183   This font family consists of the following NFSS series/shapes:\\
184   \g_@@_defined_shapes_tl
185 }
186 \@@_msg_new:nnn {fontspec} {no-font-shape}
187 {

```



```

188   Could not resolve font "#1" (it probably doesn't exist).
189   }
190   \@_msg_new:nnn {fontspec} {set-scale}
191   {
192     \l_fontspec_fontname_tl\space scale = \l_@_scale_tl.
193   }
194   \@_msg_new:nnn {fontspec} {setup-math}
195   {
196     Adjusting the maths setup (use [no-math] to avoid this).
197   }
198   \@_msg_new:nnn {fontspec} {no-script}
199   {
200     Font "#1" does not contain requested Script "#2".
201   }
202   \@_msg_new:nnn {fontspec} {opa-twice}
203   {
204     Opacity set twice, in both Colour and Opacity.\@
205     Using specification "Opacity=#1".
206   }
207   \@_msg_new:nnn {fontspec} {opa-twice-col}
208   {
209     Opacity set twice, in both Opacity and Colour.\@
210     Using an opacity specification in hex of "#1/FF".
211   }
212   \@_msg_new:nnn {fontspec} {bad-colour}
213   {
214     Bad colour declaration "#1".
215     Colour must be one of:\@
216     * a named xcolor colour\@
217     * a six-digit hex colour RRGGBB\@
218     * an eight-digit hex colour RRGGBBTT with opacity
219   }
220   Reset 'space' behaviour:
221   \char_set_catcode_ignore:n {32}

```

File V

fontspec-code-opening.dtx

1 Opening code

1.1 Package options

```
1 \DeclareOption{cm-default}
2 {
3   \@@_warning:n {cm-default-obsolete}
4 }
5 \DeclareOption {math}      { \bool_gset_true:N \g_@@_math_bool }
6 \DeclareOption {no-math}  { \bool_gset_false:N \g_@@_math_bool }
7 \DeclareOption {config}   { \bool_gset_true:N \g_@@_cfg_bool }
8 \DeclareOption {no-config}{ \bool_gset_false:N \g_@@_cfg_bool }
9 \DeclareOption {euenc}    { \bool_gset_true:N \g_@@_euenc_bool }
10 \DeclareOption {tuenc}   { \bool_gset_false:N \g_@@_euenc_bool }
11 \DeclareOption {quiet}
12 {
13   \msg_redirect_module:nnn { fontspec } { warning } { info }
14   \msg_redirect_module:nnn { fontspec } { info } { none }
15 }
16 \DeclareOption{silent}
17 {
18   \msg_redirect_module:nnn { fontspec } { warning } { none }
19   \msg_redirect_module:nnn { fontspec } { info } { none }
20 }
21 \ExecuteOptions{config,math,tuenc}
22 \ProcessOptions*
```

1.2 Encodings

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

```
23 \bool_if:NF \g_@@_euenc_bool
24 {
25   \file_if_exist:nTF {tuenc.def}
26   {
27     \cs_if_exist:cF {T@TU}
28     {
29       \@@_warning:n {tu-clash}
30       \DeclareFontEncoding{TU}{}{}
31       \DeclareFontSubstitution{TU}{lmr}{m}{n}
32     }
33   }
34   {
35     \@@_warning:n {tu-missing}
36     \bool_gset_true:N \g_@@_euenc_bool
37   }
38 }
```

```

39 \bool_if:NTF \g_@@_euenc_bool
40 {
41 <XE> \tl_gset:Nn \g_fontspec_encoding_tl {EU1}
42 <LU> \tl_gset:Nn \g_fontspec_encoding_tl {EU2}
43 }
44 { \tl_gset:Nn \g_fontspec_encoding_tl { TU } }
45 \tl_set:Nn \rmdefault {lmr}
46 \tl_set:Nn \sfdefault {lmss}
47 \tl_set:Nn \ttdefault {lmtt}
48 \RequirePackage[\g_fontspec_encoding_tl]{fontenc}
49 \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:

```

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

```

Dealing with a couple of the problems introduced by babel:

```

51 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
52 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
53 \AtBeginDocument
54 {
55 \tl_set_eq:NN \cyrillicencoding \g_fontspec_encoding_tl
56 \tl_set_eq:NN \latinencoding \g_fontspec_encoding_tl
57 }

```

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.

```

58 \bool_if:NT \g_@@_euenc_bool
59 {
60 <LU> \cs_set_eq:NN \fontspec_tmp: \XeTeXpicfile
61 <LU> \cs_set:Npn \XeTeXpicfile {}
62 \RequirePackage{xunicode}
63 <LU> \cs_set_eq:NN \XeTeXpicfile \fontspec_tmp:
64 }

```

1.3 Generic functions

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

```

\FontspecSetCheckBoolTrue
\FontspecSetCheckBoolFalse

```

```

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

```

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

```

\@@_keys_set_known:nnN

```

```

67 \cs_new:Nn \@@_keys_set_known:nnN
68 {
69 <debug> \typeout{::: Keys-set:~{#1}~{#2} }
70 \keys_set_known:nnN {#1} {#2} #3
71 <debug> \typeout{::: Leftover:~{#3} }
72 }
73 \cs_generate_variant:Nn \@@_keys_set_known:nnN {nx}

```

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

\@@_int_mult_truncate:Nn Missing in expl3, IMO.

```
74 \cs_new:Nn \@@_int_mult_truncate:Nn
75 {
76   \int_set:Nn #1 { \_dim_eval:w #2 #1 \_dim_eval_end: }
77 }
```

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

```
\@@_lua_function:ne
\@@_lua_function:nee 78 <*LU>
\@@_lua_function:neee 79 \cs_set:Npn \@@_lua_function:ne #1#2 { \lua_now:e { fontspec.#1 ("#2")
\@@_lua_function:neeee 80 \cs_set:Npn \@@_lua_function:nee #1#2#3 { \lua_now:e { fontspec.#1 ("#2", "#3")
81 \cs_set:Npn \@@_lua_function:neee #1#2#3#4 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4")
82 \cs_set:Npn \@@_lua_function:neeee #1#2#3#4#5 { \lua_now:e { fontspec.#1 ("#2", "#3", "#4", "#5"
83 </LU>
```

(End definition for \@@_lua_function:ne and others. These functions are documented on page ??.)

1.4 expl3 variants

```
84 \cs_generate_variant:Nn \int_set:Nn {Nv}
85 \cs_generate_variant:Nn \keys_set:nn {nx}
86 \cs_generate_variant:Nn \keys_set_known:nnN {nx}
87 \cs_generate_variant:Nn \prop_put:Nnn {Nxx}
88 \cs_generate_variant:Nn \prop_put:Nnn {NxV}
89 \cs_generate_variant:Nn \prop_gput_if_new:Nnn {NxV}
90 \cs_generate_variant:Nn \prop_gput:Nnn {NxN}
91 \cs_generate_variant:Nn \prop_get:NnNT {NxN}
92 \cs_generate_variant:Nn \prop_get:NnNTF {NxN}
93 \cs_generate_variant:Nn \str_if_eq:nnTF {nv}
94 \cs_generate_variant:Nn \tl_if_empty_p:n {e}
95 \cs_generate_variant:Nn \tl_if_empty:nTF {x}
96 \cs_generate_variant:Nn \tl_if_empty:nF {x}
97 \cs_generate_variant:Nn \tl_if_empty:nF {f}
98 \cs_generate_variant:Nn \tl_if_eq:nnT {ox}
99 \cs_generate_variant:Nn \tl_replace_all:Nnn {Nnx}
```

File VI

fontspec-code-fontload.dtx

1 expl3 interface for primitive font loading

\@@_primitive_font_set:Nnn

\@@_primitive_font_gset:Nnn

```
1 \cs_set:Npn \@@_primitive_font_set:Nnn #1#2#3
2 {
3   \font #1 = #2 ~at~ \dim_eval:n {#3} \scan_stop:
4 }
5 \cs_set:Npn \@@_primitive_font_gset:Nnn #1#2#3
6 {
7   \global \font #1 = #2 ~at~ \dim_eval:n {#3} \scan_stop:
8 }
```

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

\@@_font_suppress_not_found_error:

```
9 \cs_set:Npn \@@_font_suppress_not_found_error:
10 {
11   \int_set:Nn \suppressfontnotfounderror {1}
12 }
```

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

\@@_primitive_font_if_null_p:N

\@@_primitive_font_if_null:NTF

```
13 \prg_set_conditional:Nnn \@@_primitive_font_if_null:N {p,TF,T,F}
14 {
15   \ifx #1 \nullfont
16     \prg_return_true:
17   \else
18     \prg_return_false:
19   \fi
20 }
```

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

\@@_primitive_font_set_p:NnnTF

\@@_primitive_font_set:NnnTFTF

\@@_primitive_font_gset_p:NnnTF

\@@_primitive_font_gset:NnnTFTF

```
21 \prg_set_conditional:Nnn \@@_primitive_font_set:Nnn {TF,T,F}
22 {
23   \@@_primitive_font_set:Nnn #1 {#2} {#3}
24   \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
25 }
26 \prg_set_conditional:Nnn \@@_primitive_font_gset:Nnn {TF,T,F}
27 {
28   \@@_primitive_font_gset:Nnn #1 {#2} {#3}
29   \@@_primitive_font_if_null:NTF #1 {\prg_return_false:} {\prg_return_true:}
30 }
31 \cs_set:Npn \@@_primitive_font_set:Onn { \exp_last_unbraced:No \@@_primitive_font_set:Nnn }
```

```

32 \cs_set:Npn \@@_primitive_font_set:NnnF { \exp_last_unbraced:No \@@_primitive_font_set:NnnF
33 \cs_set:Npn \@@_primitive_font_gset:Onn { \exp_last_unbraced:No \@@_primitive_font_gset:Nnn
34 \cs_set:Npn \@@_primitive_font_gset:OnnF { \exp_last_unbraced:No \@@_primitive_font_gset:NnnF

```

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

\@@_primitive_font_if_exist:nTF

```

35 \prg_set_conditional:Nnn \@@_primitive_font_if_exist:n {TF,T,F}
36 {
37   \group_begin:
38   \@@_font_suppress_not_found_error:
39   \@@_primitive_font_set:Nnn \l_@@_primitive_font_{#1} { \f@size pt - 1sp }
40   \@@_primitive_font_if_null:NTF \l_@@_primitive_font
41   { \group_end: \prg_return_false: }
42   { \group_end: \prg_return_true: }
43 }

```

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

\@@_primitive_font_glyph_if_exist:NnTF

```

44 \prg_new_conditional:Nnn \@@_primitive_font_glyph_if_exist:Nn {p,TF,T,F}
45 {
46   \tex_iffontchar:D #1 #2 \scan_stop:
47   \prg_return_true:
48   \else:
49   \prg_return_false:
50   \fi:
51 }

```

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

\@@_primitive_font_set_hyphenchar:Nn

```

52 \cs_new:Nn \@@_primitive_font_set_hyphenchar:Nn
53 {
54   \tex_hyphenchar:D #1 = #2 \scan_stop:
55 }

```

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

\@@_primitive_font_get_name:N

\@@_primitive_font_current_name:

```

56 \cs_new_eq:NN \@@_primitive_font_get_name:N \fontname
57 \cs_new:Npn \@@_primitive_font_current_name:
58 {
59   \@@_primitive_font_get_name:N \tex_font:D
60 }

```

(End definition for \@@_primitive_font_get_name:N and \@@_primitive_font_current_name:. These functions are documented on page ??.)

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 27](#).

```
1 \NewDocumentCommand \fontspec { 0{} m 0{} }
2 {
3   \@@_main_fontspec:nn {#1,#3} {#2}
4   \ignorespaces
5 }
6 \NewDocumentCommand \setmainfont { 0{} m 0{} }
7 {
8   \@@_main_setmainfont:nn {#1,#3} {#2}
9   \ignorespaces
10 }
11 \NewDocumentCommand \setsansfont { 0{} m 0{} }
12 {
13   \@@_main_setsansfont:nn {#1,#3} {#2}
14   \ignorespaces
15 }
16 \NewDocumentCommand \setmonofont { 0{} m 0{} }
17 {
18   \@@_main_setmonofont:nn {#1,#3} {#2}
19   \ignorespaces
20 }
21 \NewDocumentCommand \setmathrm { 0{} m 0{} }
22 {
23   \@@_main_setmathrm:nn {#1,#3} {#2}
24 }
25 \NewDocumentCommand \setboldmathrm { 0{} m 0{} }
26 {
27   \@@_main_setboldmathrm:nn {#1,#3} {#2}
28 }
29 \NewDocumentCommand \setmathsf { 0{} m 0{} }
30 {
31   \@@_main_setmathsf:nn {#1,#3} {#2}
32 }
33 \NewDocumentCommand \setmathtt { 0{} m 0{} }
34 {
35   \@@_main_setmathtt:nn {#1,#3} {#2}
36 }
```

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

```

37 \NewDocumentCommand \setromanfont { 0{} m 0{} }
38 {
39   \@@_main_setmainfont:nn {#1,#3} {#2}
40 }

```

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

```

41 \NewDocumentCommand \newfontfamily { m 0{} m 0{} }
42 {
43   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
44 }

45 \NewDocumentCommand \renewfontfamily { m 0{} m 0{} }
46 {
47   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
48 }

49 \NewDocumentCommand \setfontfamily { m 0{} m 0{} }
50 {
51   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
52 }

53 \NewDocumentCommand \providefontfamily { m 0{} m 0{} }
54 {
55   \@@_main_newfontfamily:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
56 }

57 \NewDocumentCommand \newfontface { m 0{} m 0{} }
58 {
59   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \NewDocumentCommand
60 }

61 \NewDocumentCommand \renewfontface { m 0{} m 0{} }
62 {
63   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \RenewDocumentCommand
64 }

65 \NewDocumentCommand \setfontface { m 0{} m 0{} }
66 {
67   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \DeclareDocumentCommand
68 }

69 \NewDocumentCommand \providefontface { m 0{} m 0{} }
70 {
71   \@@_main_newfontface:NnnN #1 {#2,#4} {#3} \ProvideDocumentCommand
72 }

```

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

```

73 \NewDocumentCommand \defaultfontfeatures { t+ o m }
74 {
75   \IfNoValueTF {#2}
76     { \@@_set_default_features:nn {#1} {#3} }
77     { \@@_set_font_default_features:nnn {#1} {#2} {#3} }

```



```

78     \ignorespaces
79   }

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

80 \NewDocumentCommand \addfontfeatures {m}
81   {
82     \@@_main_addfontfeatures:n {#1}
83   }

84 \NewDocumentCommand \addfontfeature {m}
85   {
86     \@@_main_addfontfeatures:n {#1}
87   }

88 \NewDocumentCommand \newfontfeature {mm}
89   {
90     \@@_main_newfontfeature:nn {#1} {#2}
91   }

92 \NewDocumentCommand \newAATfeature {mmmm}
93   {
94     \@@_main_newAATfeature:nnnn {#1} {#2} {#3} {#4}
95   }

96 \NewDocumentCommand \newopentypefeature {mmm}
97   {
98     \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
99   }

```

`\newICUfeature` **Deprecated.**

```

100 \NewDocumentCommand \newICUfeature {mmm}
101   {
102     \@@_main_newopentypefeature:nnn {#1} {#2} {#3}
103   }

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

104 \NewDocumentCommand \aliasfontfeature {mm}
105   {
106     \@@_main_aliasfontfeature:nn {#1} {#2}
107   }

108 \NewDocumentCommand \aliasfontfeatureoption {mmm}
109   {
110     \@@_main_aliasfontfeatureoption:nnn {#1} {#2} {#3}
111   }

```

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

```

112 \NewDocumentCommand \newfontscript {mm}
113   {
114     \fontspec_new_script:nn {#1} {#2}
115   }

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

```

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

```
116 \NewDocumentCommand \newfontlanguage {mm}
117   {
118     \fontspec_new_lang:nn {#1} {#2}
119   }
```

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

```
120 \NewDocumentCommand \DeclareFontExtensions {m}
121   {
122     \@@_main_DeclareFontExtensions:n {#1}
123   }
124 \NewDocumentCommand \IfFontFeatureActiveTF {mmm}
125   {
126     \@@_main_IfFontFeatureActiveTF:nnn {#1} {#2} {#3}
127   }
```

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

```
128 \AtBeginDocument
129   {
130     \RenewDocumentCommand \oldstylenums {m}
131       {
132         \@@_main_oldstylenums:n {#1}
133       }
134   }
```

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

`\liningnums`

```
135 \NewDocumentCommand \liningnums {m}
136   {
137     \@@_main_liningnums:n {#1}
138   }
```

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

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.

```
1 \cs_new:Nn \@@_main_fontspec:nn
2 {
3   \fontspec_set_family:Nnn \f@family {#1} {#2}
4   \fontencoding { \g_@@_nfss_enc_tl }
5   \selectfont
6 }
```

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

`\rmfamily` Add an encoding switch to the three family commands.

```
\sffamily
\ttfamily
7 \cs_if_exist:NTF \@rmfamilyhook
8 {
9   \tl_put_right:Nn \@rmfamilyhook {\fontencoding \l_@@_rmfamily_encoding_tl}
10  \tl_put_right:Nn \@sffamilyhook {\fontencoding \l_@@_sffamily_encoding_tl}
11  \tl_put_right:Nn \@ttfamilyhook {\fontencoding \l_@@_ttfamily_encoding_tl}
12 }
13 {
14   \tl_replace_all:cnn { rfamily~ } { \fontfamily }
15   { \fontencoding \l_@@_rmfamily_encoding_tl \fontfamily }
16   \tl_replace_all:cnn { sffamily~ } { \fontfamily }
17   { \fontencoding \l_@@_sffamily_encoding_tl \fontfamily }
18   \tl_replace_all:cnn { ttfamily~ } { \fontfamily }
19   { \fontencoding \l_@@_ttfamily_encoding_tl \fontfamily }
20 }
21 \tl_set:Nn \l_@@_rmfamily_encoding_tl { \encodingdefault }
22 \tl_set:Nn \l_@@_sffamily_encoding_tl { \encodingdefault }
23 \tl_set:Nn \l_@@_ttfamily_encoding_tl { \encodingdefault }
```

(End 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.

```
24 \cs_new:Nn \@@_main_setmainfont:nn
25 {
26   \ifdefined\DeclareFontSeriesDefault
27     \DeclareFontSeriesDefault[rm]{bf}{\bfdefault}
28   \fi
29   \fontspec_set_family:Nnn \l_@@_rmfamily_family_tl {#1} {#2}
30   \tl_set_eq:NN \rmdefault \l_@@_rmfamily_family_tl
```

```

31 \tl_set_eq:NN \l_@@_rmfamily_encoding_tl \g_@@_nfss_enc_tl
32 \str_if_eq:eeT {\familydefault} {\rmdefault}
33 { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
34 \@@_setmainfont_hook:nn {#1} {#2} % for unicode-math only
35 \normalfont
36 }

```

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

`\setsansfont` Same as above.

```

37 \cs_new:Nn \@@_main_setsansfont:nn
38 {
39   \ifdefined\DeclareFontSeriesDefault
40     \DeclareFontSeriesDefault[sf]{bf}{\bfdefault}
41     \fi
42   \fontspec_set_family:Nnn \l_@@_sffamily_family_tl {#1} {#2}
43   \tl_set_eq:NN \sfdefault \l_@@_sffamily_family_tl
44   \tl_set_eq:NN \l_@@_sffamily_encoding_tl \g_@@_nfss_enc_tl
45   \str_if_eq:eeT {\familydefault} {\sfdefault}
46   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
47   \@@_setsansfont_hook:nn {#1} {#2} % for unicode-math only
48   \normalfont
49 }

```

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

`\setmonofont` Same as above.

```

50 \cs_new:Nn \@@_main_setmonofont:nn
51 {
52   \ifdefined\DeclareFontSeriesDefault
53     \DeclareFontSeriesDefault[tt]{bf}{\bfdefault}
54     \fi
55   \fontspec_set_family:Nnn \l_@@_ttfamily_family_tl {#1} {#2}
56   \tl_set_eq:NN \ttdefault \l_@@_ttfamily_family_tl
57   \tl_set_eq:NN \l_@@_ttfamily_encoding_tl \g_@@_nfss_enc_tl
58   \str_if_eq:eeT {\familydefault} {\ttdefault}
59   { \tl_set_eq:NN \encodingdefault \g_@@_nfss_enc_tl }
60   \@@_setmonofont_hook:nn {#1} {#2} % for unicode-math only
61   \normalfont
62 }

```

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

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

```

63 \cs_new:Nn \@@_main_setmathrm:nn
64 {
65   <XE> \fontspec_gset_family:Nnn \g_@@_mathrm_tl {#1} {#2}
66   <LU> \fontspec_gset_family:Nnn \g_@@_mathrm_tl {Renderer=Basic,#1} {#2}
67   \@@_setmathrm_hook:nn {#1} {#2} % for unicode-math only
68 }

```

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

`\setboldmathrm`

```
69 \cs_new:Nn \@@_main_setboldmathrm:nn
70 {
71 <XE> \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {#1} {#2}
72 <LU> \fontspec_gset_family:Nnn \g_@@_bfmathrm_tl {Renderer=Basic,#1} {#2}
73   \@@_setboldmathrm_hook:nn {#1} {#2} % for unicode-math only
74 }
```

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

`\setmathsf`

```
75 \cs_new:Nn \@@_main_setmathsf:nn
76 {
77 <XE> \fontspec_gset_family:Nnn \g_@@_mathsf_tl {#1} {#2}
78 <LU> \fontspec_gset_family:Nnn \g_@@_mathsf_tl {Renderer=Basic,#1} {#2}
79   \@@_setmathsf_hook:nn {#1} {#2} % for unicode-math only
80 }
```

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

`\setmathtt`

```
81 \cs_new:Nn \@@_main_setmathtt:nn
82 {
83 <XE> \fontspec_gset_family:Nnn \g_@@_mathtt_tl {#1} {#2}
84 <LU> \fontspec_gset_family:Nnn \g_@@_mathtt_tl {Renderer=Basic,#1} {#2}
85   \@@_setmathtt_hook:nn {#1} {#2} % for unicode-math only
86 }
```

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

Hooks:

```
87 \cs_set_eq:NN \@@_setmainfont_hook:nn \use_none:nn
88 \cs_set_eq:NN \@@_setsansfont_hook:nn \use_none:nn
89 \cs_set_eq:NN \@@_setmonofont_hook:nn \use_none:nn
90 \cs_set_eq:NN \@@_setmathrm_hook:nn \use_none:nn
91 \cs_set_eq:NN \@@_setmathsf_hook:nn \use_none:nn
92 \cs_set_eq:NN \@@_setmathtt_hook:nn \use_none:nn
93 \cs_set_eq:NN \@@_setboldmathrm_hook:nn \use_none:nn
```

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

```
94 \onlypreamble\setmathrm
95 \onlypreamble\setboldmathrm
96 \onlypreamble\setmathsf
97 \onlypreamble\setmathtt
```

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

```
98 \tl_gset:Nn \g_@@_mathrm_tl {\rmdefault}
99 \tl_gset:Nn \g_@@_mathsf_tl {\sfdefault}
100 \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).

```

101 \cs_new:Nn \@@_main_newfontfamily:NnnN
102   {
103     \fontspec_set_family:cnn { l_@@_ \cs_to_str:N #1 _family_tl } {#2} {#3}
104     \use:x
105     {
106       \exp_not:N #4 \exp_not:N #1 {}
107       {
108         \exp_not:N \fontfamily { \use:c { l_@@_ \cs_to_str:N #1 _family_tl } }
109         \exp_not:N \fontencoding { \g_@@_nfss_enc_tl }
110         \exp_not:N \selectfont
111       }
112     }
113   }

```

(End 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.

```

114 \cs_new:Nn \@@_main_newfontface:NnnN
115   {
116     \@@_main_newfontfamily:NnnN #1 { BoldFont={},ItalicFont={},SmallCapsFont={},#2 } {#3} #4
117   }

```

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

1.2 Font feature selection

`\@@_set_default_features:nn`

```

118 \cs_new:Nn \@@_set_default_features:nn
119   {
120     \IfBooleanTF {#1} \clist_gput_right:Nn \clist_gset:Nn
121       \g_@@_default_fontopts_clist {#2}
122   }

```

(End 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.

```

123 \cs_new:Nn \@@_set_font_default_features:nnn
124   {
125     <debug> \typeout{\unexpanded{\set_font_default_features:nnn:{#1}{#2}{#3}}}
126     \clist_map_inline:nn {#2}
127     {
128       \tl_if_single:nTF {##1}
129       { \tl_set:No \l_@@_tmp_tl { \cs:w l_@@_ \cs_to_str:N ##1 _family_tl\cs_end: } }
130       { \@@_sanitise_fontname:Nn \l_@@_tmp_tl {##1} }
131     }

```

```

132     \IfBooleanTF {#1}
133     {
134         \prop_get:NVNF \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
135         { \tl_clear:N \l_@@_tmpb_tl }
136         \tl_put_right:Nn \l_@@_tmpb_tl {#3,}
137         \prop_gput:NVV \g_@@_fontopts_prop \l_@@_tmp_tl \l_@@_tmpb_tl
138     }
139     {
140         \tl_if_empty:nTF {#3}
141         { \prop_gremove:NV \g_@@_fontopts_prop \l_@@_tmp_tl }
142         { \prop_gput:NVn \g_@@_fontopts_prop \l_@@_tmp_tl {#3,} }
143     }
144 }
145 }

```

(End 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.

```

146 \cs_new:Nn \@@_main_addfontfeatures:n
147 {
148 <debug> \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::^^J: addfontfeatures}
149   \fontspec_if_fontspec_font:TF
150   {
151     \group_begin:
152     \keys_set_known:nnN {fontspec-addfeatures} {#1} \l_@@_tmp_tl
153     \prop_get:cnN {g_@@_fontinfo_ \f@family_prop} {options} \l_@@_options_tl
154     \prop_get:cnN {g_@@_fontinfo_ \f@family_prop} {fontname} \l_@@_fontname_tl
155     \bool_set_true:N \l_@@_disable_defaults_bool
156 <debug> \typeout{ \@@_select_font_family:nn { \l_@@_options_tl , #1 } {\l_@@_fontname_tl} }
157     \use:x
158     {
159       \@@_select_font_family:nn
160       { \l_@@_options_tl , #1 } {\l_@@_fontname_tl}
161     }
162     \group_end:
163     \fontfamily \g_@@_nfss_family_tl \selectfont
164   }
165   {
166     \@@_warning:nx {addfontfeatures-ignored} {#1}
167   }

```

```

168     \ignorespaces
169   }

```

(End 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.

```

170 \cs_new:Nn \@@_main_newfontfeature:nn
171   {
172     \keys_define:nn { fontspec }
173       {
174         #1 .code:n = { \@@_update_featstr:n {#2} }
175       }
176   }

```

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

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

```

177 \cs_new:Nn \@@_main_newAATfeature:nnnn
178   {
179     \keys_if_exist:nnF { fontspec } {#1}
180       { \@@_define_aat_feature_group:n {#1} }
181
182     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
183       { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
184
185     \@@_define_aat_feature:nnnn {#1}{#2}{#3}{#4}
186   }

```

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

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

```

187 \cs_new:Nn \@@_main_newopentypefeature:nnn
188   {
189     \keys_if_exist:nnF { fontspec / options } {#1}
190       { \@@_define_opentype_feature_group:n {#1} }
191
192     \keys_if_choice_exist:nnnT {fontspec} {#1} {#2}
193       { \@@_warning:nxx {feature-option-overwrite} {#1} {#2} }
194
195     \exp_args:Nnnx \@@_define_opentype_feature:nnnnn
196       {#1} {#2} { \@@_strip_plus_minus:n {#3} } {#3} {}
197   }

```



```

198 \cs_new:Nn \@@_strip_plus_minus:n { \@@_strip_plus_minus_aux:Nq #1 \q_nil }
199 \cs_new:Npn \@@_strip_plus_minus_aux:Nq #1#2 \q_nil
200 {
201   \str_case:nnF {#1} { {+} {#2} {-} {#2} } {#1#2}
202 }

```

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

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

```

203 \cs_new:Nn \@@_main_aliasfontfeature:nn
204 {
205   <debug> \typeout{::::::::::::::::::::::::::^~J:: aliasfontfeature{#1}{#2}}
206   \bool_set_false:N \l_@@_alias_bool
207
208   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
209   {
210     \keys_if_exist:nnT {##1} {#1}
211     {
212       <debug> \typeout{:::: Key~exists~##1~/~#1}
213       \bool_set_true:N \l_@@_alias_bool
214       \keys_define:nn {##1}
215         { #2 .code:n = { \keys_set:nn {##1} { #1 = {####1} } } }
216     }
217   }
218
219   \bool_if:NF \l_@@_alias_bool
220   { \@@_warning:nx {rename-feature-not-exist} {#1} }
221 }

```

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

\aliasfontfeatureoption

```

222 \cs_new:Nn \@@_main_aliasfontfeatureoption:nnn
223 {
224   \bool_set_false:N \l_@@_alias_bool
225
226   \clist_map_inline:Nn \g_@@_all_keyval_modules_clist
227   {
228     \keys_if_exist:nnT { ##1 / #1 } {#2}
229     {
230       <debug> \typeout{:::: Keyval~exists~##1~/~#1~::~#2}
231       \bool_set_true:N \l_@@_alias_bool
232       \keys_define:nn { ##1 / #1 }
233         { #3 .code:n = { \keys_set:nn {##1} { #1 = {#2} } } }
234     }
235
236     \keys_if_exist:nnT { ##1 / #1 } {#2Reset}
237     {
238       <debug> \typeout{:::: Keyval~exists~##1~/~#1~::~#2Reset}
239       \keys_define:nn { ##1 / #1 }
240         { #3Reset .code:n = { \keys_set:nn {##1} { #1 = {#2Reset} } } }
241     }

```

```

242
243     \keys_if_exist:nnT { ##1 / #1 } {#20ff}
244     {
245 <debug> \typeout{::: Keyval~exists~##1~/~#1~/~#20ff}
246         \keys_define:nn { ##1 / #1 }
247             { #30ff .code:n = { \keys_set:nn {##1} { #1 = {#20ff} } } }
248     }
249 }
250
251 \bool_if:NF \l_@@_alias_bool
252 { \@@_warning:nx {rename-feature-not-exist} {#1/#2} }
253 }

```

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

\@@_main_DeclareFontExtensions:n

```

254 \cs_new:Nn \@@_main_DeclareFontExtensions:n
255 {
256     \clist_set:Nn \l_@@_extensions_clist { #1 }
257 }

```

Defaults:

```

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

```

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

1.4 High level conditionals

\IfFontFeatureActiveTF

```

259 \cs_new:Nn \@@_main>IfFontFeatureActiveTF:nnn
260 {
261 <debug> \typeout{^^J::::::::::::::::::::::::::::::::::::::::::::::::::}
262 <debug> \typeout{:IfFontFeatureActiveTF \exp_not:n{#1}{#2}{#3}}
263     \@@_if_font_feature:nTF {#1} {#2} {#3}
264 }
265 \prg_new_conditional:Nnn \@@_if_font_feature:n {TF}
266 {
267     \tl_gclear:N \g_@@_single_feat_tl
268     \group_begin:
269         \@@_font_suppress_not_found_error:
270         \@@_init:
271         \bool_set_true:N \l_@@_ot_bool
272         \bool_set_true:N \l_@@_never_check_bool
273         \bool_set_false:N \l_@@_firsttime_bool
274         \clist_clear:N \l_@@_fontfeat_clist
275         \@@_get_features:n {#1}
276     \group_end:
277
278 <debug> \typeout{:::> \exp_not:N\g_@@_rawfeatures_sclist->~{\g_@@_rawfeatures_sclist}}
279 <debug> \typeout{:::> \exp_not:N\g_@@_single_feat_tl->~{\g_@@_single_feat_tl}}
280
281     \tl_if_empty:NTF \g_@@_single_feat_tl { \prg_return_false: }

```

```

282     {
283       \exp_args:NV \fontspec_if_current_feature:nTF \g_@@_single_feat_tl
284       { \prg_return_true: } { \prg_return_false: }
285     }
286   }

```

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

1.5 \oldstylenums and \liningnums

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

```

\liningnums 287 \cs_new_protected:Nn \@@_main_oldstylenums:n
288   {
289     \group_begin:
290     \addfontfeature{Numbers=OldStyle}
291     #1
292     \group_end:
293   }
294 \cs_new_protected:Nn \@@_main_liningnums:n
295   {
296     \group_begin:
297     \addfontfeature{Numbers=Lining}
298     #1
299     \group_end:
300   }

```

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

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.)

```
\fontspec_if_fontspec_font:TF Test whether the currently selected font has been loaded by fontspec.
1 \prg_new_conditional:Nnn \fontspec_if_fontspec_font: {TF,T,F}
2 {
3   \cs_if_exist:cTF {g_@@_fontinfo_ \f@family _prop} \prg_return_true: \prg_return_false:
4 }
```

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

```
\fontspec_if_aat_feature:nnTF Conditional to test if the currently selected font contains the AAT feature (#1,#2).
5 \prg_new_conditional:Nnn \fontspec_if_aat_feature:nn {TF,T,F}
6 {
7   \fontspec_if_fontspec_font:TF
8   {
9     \@@_set_font_type:N \font
10    \bool_if:NTF \l_@@_atsui_bool
11    {
12      \@@_make_AAT_feature_string:NnnTF \font {#1} {#2}
13      \prg_return_true: \prg_return_false:
14    }
15    {
16      \prg_return_false:
17    }
18  }
19  {
20    \prg_return_false:
21  }
22 }
```

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

```
\fontspec_if_opentype:TF Test whether the currently selected font is an OpenType font. Always true for LuaTeX fonts.
23 \prg_new_conditional:Nnn \fontspec_if_opentype: {TF,T,F}
24 {
25   \fontspec_if_fontspec_font:TF
26   {
```

```

27     \@@_set_font_type:N \font
28     \bool_if:NTF \l_@@_ot_bool \prg_return_true: \prg_return_false:
29   }
30   {
31     \prg_return_false:
32   }
33 }

```

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

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

```

34 \prg_new_conditional:Nnn \fontspec_if_feature:n {TF,T,F}
35 {
36   \fontspec_if_fontspec_font:TF
37   {
38     \@@_set_font_type:N \font
39     \bool_if:NTF \l_@@_ot_bool
40     {
41       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmp_tl
42       \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
43
44       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-num} \l_@@_tmp_tl
45       \int_set:Nn \l_@@_language_int {\l_@@_tmp_tl}
46
47       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_script_tl
48       \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_@@_lang_tl
49
50       \@@_check_ot_feat:NnTF \font {#1} {\prg_return_true:} {\prg_return_false:}
51     }
52     {
53       \prg_return_false:
54     }
55   }
56   {
57     \prg_return_false:
58   }
59 }

```

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

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

```

60 \prg_new_conditional:Nnn \fontspec_if_feature:nnn {TF,T,F}
61 {
62   \fontspec_if_fontspec_font:TF
63   {
64     \@@_set_font_type:N \font
65     \bool_if:NTF \l_@@_ot_bool
66     {

```

```

67         \@@_check_ot_feat:NnnTF \font {#3} {#2} {#1} \prg_return_true: \prg_return_false
68     }
69     { \prg_return_false: }
70 }
71 { \prg_return_false: }
72 }

```

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

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

```

73 \prg_new_conditional:Nnn \fontspec_if_script:n {TF,T,F}
74 {
75     \fontspec_if_fontspec_font:TF
76     {
77         \@@_set_font_type:N \font
78         \bool_if:NTF \l_@@_ot_bool
79         {
80             \@@_check_script:NnTF \font {#1} \prg_return_true: \prg_return_false:
81         }
82         { \prg_return_false: }
83     }
84     { \prg_return_false: }
85 }

```

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

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

```

86 \prg_new_conditional:Nnn \fontspec_if_language:n {TF,T,F}
87 {
88     \fontspec_if_fontspec_font:TF
89     {
90         \@@_set_font_type:N \font
91         \bool_if:NTF \l_@@_ot_bool
92         {
93             \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-num} \l_@@_tmp_tl
94             \int_set:Nn \l_@@_script_int {\l_@@_tmp_tl}
95             \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_script_tl
96
97             \@@_check_lang:NnTF \font {#1} \prg_return_true: \prg_return_false:
98         }
99         { \prg_return_false: }
100     }
101     { \prg_return_false: }
102 }

```

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

`\fontspec_if_language:nnTF` Test whether the currently selected font contains the raw OpenType language tag #2 in script #1. E.g.: `\fontspec_if_language:nnTF {cyr1} {SRB} {True} {False}`. Returns false if the font is not loaded by fontspec or is not an OpenType font.

```

103 \prg_new_conditional:Nnn \fontspec_if_language:nn {TF,T,F}
104   {
105     \fontspec_if_fontspec_font:TF
106     {
107       \@@_set_font_type:N \font
108       \bool_if:NTF \l_@@_ot_bool
109       {
110         \@@_check_lang:NnnTF \font {#2} {#1} \prg_return_true: \prg_return_false:
111       }
112       { \prg_return_false: }
113     }
114     { \prg_return_false: }
115   }

```

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

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

```

116 \prg_new_conditional:Nnn \fontspec_if_current_script:n {TF,T,F}
117   {
118     \fontspec_if_fontspec_font:TF
119     {
120       \@@_set_font_type:N \font
121       \bool_if:NTF \l_@@_ot_bool
122       {
123         \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {script-tag} \l_@@_tmp_tl
124         \str_if_eq:nVTF {#1} \l_@@_tmp_tl
125         {\prg_return_true:} {\prg_return_false:}
126       }
127       { \prg_return_false: }
128     }
129     { \prg_return_false: }
130   }

```

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

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

```

131 \prg_new_conditional:Nnn \fontspec_if_current_language:n {TF,T,F}
132   {
133     \fontspec_if_fontspec_font:TF
134     {
135       \@@_set_font_type:N \font
136       \bool_if:NTF \l_@@_ot_bool
137       {
138         \prop_get:cnN {g_@@_fontinfo_ \f@family _prop} {lang-tag} \l_@@_tmp_tl
139         \str_if_eq:nVTF {#1} \l_@@_tmp_tl
140         {\prg_return_true:} {\prg_return_false:}
141       }
142       { \prg_return_false: }
143     }

```

```

144     { \prg_return_false: }
145   }

```

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

```

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

```

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

We want to store the actual name of the font family within the *family* variable because the actual L^AT_EX family name is automatically generated by fontspec and it's easier to keep it that way.

```

146 \cs_new:Nn \@@_tl_new_if_free:N { \tl_if_exist:NF #1 { \tl_new:N #1 } }
147 \cs_new:Nn \@@_set_family:NnnN
148   {
149     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
150     \tl_set:Nn \l_@@_family_label_tl {#1}
151     \@@_select_font_family:nn {#2} {#3}
152     \@@_tl_new_if_free:N #1
153     #4 #1 \l_fontspec_family_tl
154     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
155   }
156 \cs_new:Nn \fontspec_gset_family:Nnn { \@@_set_family:NnnN #1 {#2} {#3} \tl_gset_eq:NN }
157 \cs_new:Nn \fontspec_set_family:Nnn { \@@_set_family:NnnN #1 {#2} {#3} \tl_set_eq:NN }
158 \cs_generate_variant:Nn \fontspec_set_family:Nnn {c}

```

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

```

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

```

```

159 \tl_new:N \l_@@_fontface_cs_tl
160 \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font}
161 \cs_new:Nn \@@_set_fontface:NNnnN
162   {
163     \tl_set:Nn \l_@@_fontface_cs_tl {#1}
164     \tl_set:Nn \l_@@_family_label_tl {#2}
165     \@@_select_font_family:nn {#3} {#4}
166     #5 #2 \l_fontspec_family_tl
167     \tl_set:Nn \l_@@_fontface_cs_tl {\l_fontspec_font} % reset
168   }
169 \cs_new:Nn \fontspec_gset_fontface:NNnnn { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_gset_eq:NN }
170 \cs_new:Nn \fontspec_set_fontface:NNnnn { \@@_set_fontface:NNnnN #1 #2 {#3} {#4} \tl_set_eq:NN }

```

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

```

\fontspec_font_if_exist:n

```

```

171 \prg_new_conditional:Nnn \fontspec_font_if_exist:n {TF,T,F}
172   {
173     \group_begin:
174     \@@_init:

```



```

175     \@@_if_detect_external:nT {#1} { \@@_font_is_file: }
176     \@@_primitive_font_if_exist:nTF { \@@_construct_font_call:nn {#1} {} }
177         { \group_end: \prg_return_true: }
178         { \group_end: \prg_return_false: }
179     }

```

```

180 \cs_set_eq:NN \IfFontExistsTF \fontspec_font_if_exist:nTF

```

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

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

```

181 \prg_new_conditional:Nnn \fontspec_if_current_feature:n {TF,T,F}
182 {
183 <debug>\typeout{:~fontspec_if_current_feature:n~{#1}}
184 <debug>\typeout{:::~~primitive_font_current_name::~~\@@_primitive_font_current_name:}
185     \exp_args:Nxx \tl_if_in:nnTF
186         { \@@_primitive_font_current_name: } { \tl_to_str:n {#1} }
187         { \prg_return_true: } { \prg_return_false: }
188 }

```

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

`\fontspec_if_small_caps:TF`

```

189 \prg_new_conditional:Nnn \fontspec_if_small_caps: {TF,T,F}
190 {
191     \@@_if_merge_shape:nTF {sc}
192     {
193         \tl_set_eq:Nc \l_@@_smcp_shape_tl { \@@_shape_merge:nn {\f@shape} {sc} }
194     }
195     {
196         \tl_set:Nn \l_@@_smcp_shape_tl {sc}
197     }
198
199     \cs_if_exist:cTF { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
200     {
201         \tl_if_eq:ccTF
202             { \f@encoding/\f@family/\f@series/\l_@@_smcp_shape_tl }
203             { \f@encoding/\f@family/\f@series/\shapedefault }
204             { \prg_return_false: }
205             { \prg_return_true: }
206     }
207     { \prg_return_false: }
208 }

```

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


```

26 \@@_load_font:
27 \@@_set_scriptlang:
28 \@@_get_features:n {}
29 \bool_set_false:N \l_@@_firsttime_bool
30
31 \@@_save_family_needed:nTF {#2}
32 {
33   \@@_save_family:nn {#1} {#2}
34 <debug>\@@_warning:nxx {defining-font} {#1} {#2}
35 }
36 {
37 <debug>\typeout{Font~ family~ already~ defined.}
38 }
39 \group_end:
40
41 \tl_set_eq:NN \l_fontspec_family_tl \g_@@_nfss_family_tl
42 }

```

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

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

```

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

```

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

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

```

44 \cs_new:Nn \@@_sanitise_fontname:Nn
45 {
46   \tl_set:Nx #1 {#2}
47 <LU> \tl_remove_all:Nn #1 {-}
48   \clist_map_inline:Nn \l_@@_extensions_clist
49   {
50     \tl_if_in:NnT #1 {##1}
51     {
52       \tl_remove_once:Nn #1 {##1}
53       \tl_set:Nn \l_@@_extension_tl {##1}
54       \clist_map_break:
55     }
56   }
57 }

```

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

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

```

58 \prg_new_conditional:Nnn \@@_if_detect_external:n {T}
59 {
60 <debug> \typeout{: : @@@_if_detect_external:n { \exp_not:n {#1} } }
61   \clist_map_inline:Nn \l_@@_extensions_clist
62   {

```

```

63     \bool_set_false:N \l_@@_tmpa_bool
64     \exp_args:Nx % <- this should be handled earlier
65     \tl_if_in:nnT {#1 <= end_of_string} {##1 <= end_of_string}
66     { \bool_set_true:N \l_@@_tmpa_bool \clist_map_break: }
67   }
68   \bool_if:NTF \l_@@_tmpa_bool \prg_return_true: \prg_return_false:
69 }

```

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

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

```

70 \cs_new:Nn \@@_init_ttc:n
71 {
72   \str_if_eq:eeT { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
73   {
74     \@@_sanitise_fontname:Nn \l_@@_fontname_it_tl {#1}
75     \@@_sanitise_fontname:Nn \l_@@_fontname_bf_tl {#1}
76     \@@_sanitise_fontname:Nn \l_@@_fontname_bfit_tl {#1}
77   }
78 }

```

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

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

```

79 \cs_new:Nn \@@_load_external_fontoptions:Nn
80 {
81   \bool_if:NT \l_@@_fontcfg_bool
82   {
83     <debug> \typeout{: @@@_load_external_fontoptions:Nn \exp_not:N #1 {#2} }
84     \@@_sanitise_fontname:Nn #1 {#2}
85     \tl_set:Nx \l_@@_ext_filename_tl {#1.fontspec}
86     \tl_remove_all:Nn \l_@@_ext_filename_tl {~}
87     \prop_if_in:NVF \g_@@_fontopts_prop #1
88     {
89       \exp_args:No \file_if_exist:nT { \l_@@_ext_filename_tl }
90       { \file_input:n { \l_@@_ext_filename_tl } }
91     }
92   }
93 }

```

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

\@@_extract_all_features:

```

94 \cs_new:Nn \@@_extract_all_features:n
95 {
96 <debug> \typeout{: @@@_extract_all_features:n { \unexpanded {#1} } }
97   \bool_if:NTF \l_@@_disable_defaults_bool
98   {
99     \clist_set:Nx \l_@@_all_features_clist {#1}
100  }

```

```

101 {
102   \prop_get:NVNF \g_@@_fontopts_prop \l_fontspec_fontname_tl \l_@@_fontopts_clist
103   { \clist_clear:N \l_@@_fontopts_clist }
104
105   \prop_get:NVNF \g_@@_fontopts_prop \l_@@_family_label_tl \l_@@_family_fontopts_clist
106   { \clist_clear:N \l_@@_family_fontopts_clist }
107   \tl_clear:N \l_@@_family_label_tl
108
109   \clist_set:Nx \l_@@_all_features_clist
110   {
111     \g_@@_default_fontopts_clist,
112     \l_@@_family_fontopts_clist,
113     \l_@@_fontopts_clist,
114     #1
115   }
116 }
117 }

```

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

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

Perform the (multi-step) feature parsing process.

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

```

118 \cs_new:Nn \@@_preparse_features:
119 {
120 <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.

```

121
122   \@@_keys_set_known:nxN {fontspec-preparse-external}
123   { \l_@@_all_features_clist }
124   \l_@@_keys_leftover_clist
125

```

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.

```

126   \tl_set_eq:NN \l_fontspec_fontname_tl \l_@@_fontname_up_tl
127   \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_keys_leftover_clist}
128   \l_@@_keys_leftover_clist
129   \@@_keys_set_known:nxN {fontspec-preparse} {\l_@@_keys_leftover_clist}
130   \l_@@_fontfeat_clist
131 }

```

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

\@@_load_font:

```

132 \cs_new:Nn \@@_load_font:
133 {

```

```

134 <debug>\typeout{:: @@_load_font}
135
136 <debug>\typeout{Set~ base~ font~ for~ preliminary~ analysis: \@@_construct_font_call:nn { \l_
137   \@@_primitive_font_set:NnnF \l_@@_test_font
138   { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
139   { \f@size pt - 2sp }
140   { \@@_error:nx {font-not-found} {\l_@@_fontname_up_tl} }
141
142 <debug>\typeout{Set~ base~ font~ properly: \@@_construct_font_call:nn { \l_@@_fontname_up_tl }
143   \@@_set_font_type:N \l_@@_test_font
144   \@@_primitive_font_gset:Onn \l_@@_fontface_cs_tl
145   { \@@_construct_font_call:nn { \l_@@_fontname_up_tl } { \l_@@_pre_feat_sclist } }
146   { \f@size pt + 2sp }
147
148   \l_@@_fontface_cs_tl % this is necessary for LuaLaTeX to check the scripts properly
149
150 }

```

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

\@@_construct_font_call:nn Constructs the complete font invocation. #1 : Base name

#2 : Extension

#3 : TTC Index

#4 : Renderer

#5 : Optical size

#6 : Font features

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

```

151 \cs_new:Nn \@@_construct_font_call:nnnnnn
152 {
153 <XE> " \@@_fontname_wrap:n { #1 #2 #3 }
154 <LU> " \@@_fontname_wrap:n { #1 #2 } #3
155   #4 #5
156   \str_if_eq:eeF {#6}{ } {:#6} "
157 }

```

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

```

158 \cs_new:Nn \@@_construct_font_call:nn
159 {
160   \@@_construct_font_call:nnnnnn
161   {#1}
162   \l_@@_extension_tl
163   \l_@@_ttc_index_tl
164   \l_@@_renderer_tl
165   \l_@@_optical_size_tl
166   {#2}
167 }

```

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

`\@@_font_is_file:` 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’. X₃TeX’s syntax is followed since `luaotfload` provides compatibility.

```

168 \cs_new:Nn \@@_font_is_name:
169   {
170     \cs_set_eq:NN \@@_fontname_wrap:n \use:n
171   }

172 \cs_new:Nn \@@_font_is_file:
173   {
174     \cs_set:Npn \@@_fontname_wrap:n ##1 { [ \l_@@_font_path_tl ##1 ] }
175   }

```

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

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

```

176 \cs_new:Nn \@@_set_scriptlang:
177   {
178     <debug> \typeout{:: _set_scriptlang:}
179     \bool_if:NT \l_@@_firsttime_bool
180       {
181         \tl_if_empty:NF \l_@@_script_name_tl
182           {
183             <debug> \typeout{::: Script=\l_@@_script_name_tl, Language=\l_@@_lang_name_tl}
184             \keys_set:nx {fontspec-opentype} {Script=\l_@@_script_name_tl}
185             \keys_set:nx {fontspec-opentype} {Language=\l_@@_lang_name_tl}
186           }
187       }
188   }

```

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

`\@@_get_features:Nn` This macro is a wrapper for `\keys_set:n` 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.

```

189 \cs_new:Nn \@@_get_features:n
190   {
191     <debug> \typeout{:: @@_get_features:Nn { \exp_not:n {#1} } }
192     \@@_init_fontface:
193     \@@_keys_set_known:nxN {fontspec-renderer} {\l_@@_fontfeat_clist,#1}
194     \l_@@_keys_leftover_clist
195     \@@_keys_set_known:nxN {fontspec} {\l_@@_keys_leftover_clist} \l_@@_keys_leftover_clist
196     <*XE>
197     \bool_if:NTF \l_@@_ot_bool
198       {
199         <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist}
200         \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
201       }
202     {
203     <debug> \typeout{::: Setting~ keys~ for~ AAT/Graphite~ font~ features::~"\l_@@_keys_leftover_

```

```

204     \bool_if:nT { \l_@@_atsui_bool || \l_@@_graphite_bool }
205     { \keys_set:nV {fontspec-aat} \l_@@_keys_leftover_clist }
206   }
207 </XE>
208 <*LU>
209 <debug> \typeout{::: Setting~ keys~ for~ OpenType~ font~ features::~"\l_@@_keys_leftover_clist
210     \keys_set:nV {fontspec-opentype} \l_@@_keys_leftover_clist
211 </LU>
212
213   \tl_if_empty:NF \l_@@_mapping_tl
214     { \@@_update_featstr:n { mapping = \l_@@_mapping_tl } }
215
216   \str_if_eq:eeF { \l_@@_hexcol_tl \l_@@_opacity_tl }
217     { \c_@@_hexcol_tl \c_@@_opacity_tl }
218     { \@@_update_featstr:n { color = \l_@@_hexcol_tl\l_@@_opacity_tl } }
219   }

```

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

`\@@_save_family_needed:nTF` Check if the family is unique and, if so, save its information. (`\addfontfeature` and other macros use this data.) Then the font family and its shapes are defined in the NFSS.

Now we have a unique (in fact, too unique!) string that contains the family name and every option in abbreviated form. This is used with a counter to create a simple NFSS family name for the font we're selecting.

```

220 \prg_new_conditional:Nnn \@@_save_family_needed:n { TF }
221   {
222
223   <debug> \typeout{save~ family::~ #1}
224   <debug> \typeout{== fontid_tl: "\l_@@_fontid_tl".}
225
226   \tl_if_empty:NTF \l_@@_nfss_fam_tl
227     {
228     \prop_get:NVNTF \g_@@_fontid_family_prop \l_@@_fontid_tl \l_@@_tmp_tl
229     {
230     \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_tmp_tl
231     \prg_return_false:
232     }
233     {
234     \tl_set:Nx \l_@@_tmp_tl {#1}
235     \tl_remove_all:Nn \l_@@_tmp_tl { ~ }
236     \@@_save_fontid_family:VV \l_@@_fontid_tl \l_@@_tmp_tl
237     \prg_return_true:
238     }
239   }
240   {
241   \tl_gset_eq:NN \g_@@_nfss_family_tl \l_@@_nfss_fam_tl
242   \cs_undefine:c { g_@@_fontinfo_ \g_@@_nfss_family_tl _prop }
243   \prg_return_true:
244   }
245 }

```



```

246 \cs_new:Nn \@@_save_fontid_family:nn
247 {
248   \prop_get:NnNTF \g_@@_family_int_prop {#2} \l_@@_tmp_tl
249   {
250     \tl_set:Nx \l_@@_tmp_tl
251     { \int_eval:n { \l_@@_tmp_tl + 1 } }
252   }
253   { \tl_set:Nn \l_@@_tmp_tl { 0 } }
254   \prop_gput:NnV \g_@@_family_int_prop {#2} \l_@@_tmp_tl
255   \tl_gset:Nx \g_@@_nfss_family_tl { #2 ( \l_@@_tmp_tl ) }
256   \prop_gput:NnV \g_@@_fontid_family_prop {#1} \g_@@_nfss_family_tl
257 }
258 \cs_generate_variant:Nn \@@_save_fontid_family:nn { VV }

```

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

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

```

259 \cs_new:Nn \@@_save_family:nn
260 {
261   \@@_save_fontinfo:n {#2}
262   \@@_find_autofonts:
263   \DeclareFontFamily{\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{-}
264   \@@_set_faces:
265   \@@_info:nxx {defining-font} {#1} {#2}
266 }

```

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

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

```

267 \cs_new:Nn \@@_save_fontinfo:n
268 {
269   \prop_new:c {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop}
270   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontname} { #1 }
271   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {options} { \l_@@_all_features }
272   \prop_gput:cnx {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {fontdef}
273   {
274     \@@_construct_font_call:nn {\l_fontspeg_fontname_tl}
275     { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
276   }
277   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-num} \l_@@_script_int
278   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-num} \l_@@_language_int
279   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {script-tag} \l_@@_script_tl
280   \prop_gput:cnV {g_@@_fontinfo_ \g_@@_nfss_family_tl _prop} {lang-tag} \l_@@_lang_tl
281 }

```

(End 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 `\bdefault` 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:`

```

282 \cs_new:Nn \@@_find_autofonts:
283 {
284   \bool_if:nF {\l_@@_noit_bool || \l_@@_nobf_bool}
285   {
286     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_it_tl} {/B}
287     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_@@_fontname_bf_tl} {/I}
288     \@@_set_autofont:Nnn \l_@@_fontname_bfit_tl {\l_fontspeg_fontname_tl} {/BI}
289   }
290
291   \bool_if:NF \l_@@_nobf_bool
292   {
293     \@@_set_autofont:Nnn \l_@@_fontname_bf_tl {\l_fontspeg_fontname_tl} {/B}
294   }
295
296   \bool_if:NF \l_@@_noit_bool
297   {
298     \@@_set_autofont:Nnn \l_@@_fontname_it_tl {\l_fontspeg_fontname_tl} {/I}
299   }
300
301   \@@_set_autofont:Nnn \l_@@_fontname_bfsl_tl {\l_@@_fontname_sl_tl} {/B}
302 }

```

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

`\@@_set_faces:`

```

303 \cs_new:Nn \@@_set_faces:
304 {
305   \@@_add_nfssfont:nmnn \mdefault \shapedefault \l_fontspeg_fontname_tl \l_@@_fontfeat_up_
306   \@@_add_nfssfont:nmnn \bfdefault \shapedefault \l_@@_fontname_bf_tl \l_@@_fontfeat_bf_
307   \@@_add_nfssfont:nmnn \mdefault \itdefault \l_@@_fontname_it_tl \l_@@_fontfeat_it_
308   \@@_add_nfssfont:nmnn \mdefault \sldefault \l_@@_fontname_sl_tl \l_@@_fontfeat_sl_
309   \@@_add_nfssfont:nmnn \bfdefault \itdefault \l_@@_fontname_bfit_tl \l_@@_fontfeat_bfi
310   \@@_add_nfssfont:nmnn \bfdefault \sldefault \l_@@_fontname_bfsl_tl \l_@@_fontfeat_bfs
311   \prop_map_inline:Nn \l_@@_nfssfont_prop { \@@_set_faces_aux:nmnn ##2 }
312 }
313 \cs_new:Nn \@@_set_faces_aux:nmnn
314 {
315   \fontspec_complete_fontname:Nn \l_@@_curr_fontname_tl {#3}
316   \@@_make_font_shapes:Nmnnn \l_@@_curr_fontname_tl {#1} {#2} {#4} {#5}
317 }

```

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

`\fontspec_complete_fontname:Nn` This macro defines #1 as the input with any * tokens of its input replaced by the font name. This lets us define supplementary fonts in full ("Baskerville Semibold") or in abbreviation ("* Semibold").

```

318 \cs_new:Nn \fontspec_complete_fontname:Nn
319 {
320   \tl_set:Nx #1 {#2}
321   \tl_replace_all:Nnx #1 {*} {\l_@@_basename_tl}
322   <LU> \tl_remove_all:Nn #1 {-}
323 }

```

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

```

\@@_add_nfssfont:nmnn #1 : series
#2 : shape
#3 : fontname
#4 : fontspec features

```

```

324 \cs_new:Nn \@@_add_nfssfont:nmnn
325 {
326   \tl_set:Nx \l_@@_this_font_tl {#3}
327
328   \tl_if_empty:xF {#4}
329   { \clist_set:Nn \l_@@_sizefeat_clist {Size={-}} }
330   { \@@_keys_set_known:nxN {fontspec-preparse-nested} {#4} \l_@@_tmp_tl }
331
332   \tl_if_empty:NF \l_@@_this_font_tl
333   {
334     \prop_put:Nxx \l_@@_nfssfont_prop {#1/#2}
335     { {#1}{#2}{\l_@@_this_font_tl}{#4}{\l_@@_sizefeat_clist} }
336   }
337 }

```

(End definition for `\@@_add_nfssfont:nmnn`. 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.

```

338 \cs_new:Nn \@@_set_font_type:N
339 {
340   <debug> \typeout{:: @@@_set_font_type:}
341   <*XE>
342   \bool_set_false:N \l_@@_tfm_bool
343   \bool_set_false:N \l_@@_atsui_bool
344   \bool_set_false:N \l_@@_ot_bool
345   \bool_set_false:N \l_@@_mm_bool
346   \bool_set_false:N \l_@@_graphite_bool
347   \ifcase\XeTeXfonttype #1
348   <debug> \typeout{::: TFM}
349   \bool_set_true:N \l_@@_tfm_bool
350   \or
351   <debug> \typeout{::: AAT}

```

```

352 \bool_set_true:N \l_@@_atsui_bool
353 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/AAT} }
354 \ifnum\XeTeXcountvariations #1 > 0\relax
355 <debug> \typeout{::: MM}
356 \bool_set_true:N \l_@@_mm_bool
357 \fi
358 \or
359 <debug> \typeout{::: OpenType}
360 \bool_set_true:N \l_@@_ot_bool
361 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/OT} }
362 \or
363 <debug> \typeout{::: Graphite}
364 \bool_set_true:N \l_@@_graphite_bool
365 \tl_if_empty:NT \l_@@_renderer_tl { \tl_set:Nn \l_@@_renderer_tl {/GR} }
366 \fi
367 </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:

```

368 <*LU>
369   \bool_set_true:N \l_@@_ot_bool
370 </LU>
371   }

```

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

```

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

```

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

We can't match external fonts in this way (in X_YTeX anyway; todo: test with LuaTeX). If `` 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 `` is empty, then proceed. If not found, `` remains empty. Otherwise, we have a match.

```

372 \cs_new:Nn \@@_set_autofont:Nnn
373   {
374     \bool_if:NF \l_@@_external_bool
375     {
376       \tl_if_empty:xF {#2}
377       {
378         \tl_if_empty:NT #1
379         {
380           \@@_if_autofont:nnTF {#2} {#3}
381           { \tl_set:Nx #1 {#2#3} }
382           { \@@_info:nx {no-font-shape} {#2#3} }
383         }
384       }
385     }

```

```

384     }
385   }
386 }
387 \prg_new_conditional:Nnn \@@_if_autofont:nn {T,TF}
388 {
389   \group_begin:
390   \@@_primitive_font_set:Nnn \l_@@_tmpa_font { \@@_construct_font_call:nn {#1} { \l_@@_pr
391   \@@_primitive_font_set:Nnn \l_@@_tmpb_font { \@@_construct_font_call:nn {#1#2} { \l_@@_pr
392   \str_if_eq:eeTF { \@@_primitive_font_get_name:N \l_@@_tmpa_font } { \@@_primitive_font_ge
393     { \group_end: \prg_return_false: }
394     { \group_end: \prg_return_true: }
395 }

```

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

```

\@@_make_font_shapes:Nnnnn #1 : Font name
                          #2 : Font series
                          #3 : Font shape
                          #4 : Font features
                          #5 : Size features

```

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

```

396 \cs_new:Nn \@@_make_font_shapes:Nnnnn
397 {
398   \group_begin:
399     \@@_keys_set_known:nxN {fontspec-prepare-external} { #4 } \l_@@_leftover_clist
400     \@@_load_fontname:Nn \l_fontspeg_fontname_tl {#1}
401     \@@_declare_shape:nxxx {#2} {#3} { \l_@@_fontopts_clist, \l_@@_leftover_clist } {#5}
402   \group_end:
403 }
404 \cs_new:Nn \@@_load_fontname:Nn
405 {
406 <debug> \typeout{:: @@@_load_fontname:Nn \exp_not:N #1 (#1) {#2} }
407   \@@_load_external_fontoptions:Nn #1 {#2}
408   \prop_get:NVNF \g_@@_fontopts_prop #1 \l_@@_fontopts_clist
409   { \clist_clear:N \l_@@_fontopts_clist }
410   \keys_set_groups:nnV {fontspec/fontname} {getfontname} \l_@@_fontopts_clist
411   \@@_primitive_font_set:OnnF \l_@@_fontface_cs_tl
412     { \@@_construct_font_call:nn {#1} { \l_@@_pre_feat_sclist } } { \f@size pt + 2sp }
413     { \@@_error:nx {font-not-found} {#2} }
414 }
415 \keys_define:nn {fontspec/fontname}
416 {
417   Font .tl_set:N = \l_fontspeg_fontname_tl ,
418   Font .groups:n = {getfontname} ,
419 }

```

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

```

\@@_declare_shape:nnnn #1 : Font series
                      #2 : Font shape

```

#3 : Font features

#4 : Size features

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

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

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

```
420 \cs_new:Nn \@@_declare_shape:nmmm
421 {
422 <debug>\typeout{=~ declare_shape:~{\l_fontspeg_fontname_tl}~{#1}~{#2}}
423 \tl_build_begin:N \l_@@_nfss_tl
424 \tl_build_begin:N \l_@@_nfss_sc_tl
425 \tl_set_eq:NN \l_@@_saved_fontname_tl \l_fontspeg_fontname_tl
426
427 \exp_args:Nx \clist_map_inline:nn {#4} { \@@_setup_single_size:nn {#3} {##1} }
428
429 \tl_build_end:N \l_@@_nfss_tl
430 \tl_build_end:N \l_@@_nfss_sc_tl
431
432 \@@_declare_shapes_normal:nn {#1} {#2}
433 \@@_declare_shapes_smcaps:nn {#1} {#2}
434 \@@_declare_shape_slanted:nn {#1} {#2}
435 \@@_declare_shapes_bx:nn {#1} {#2}
436 \@@_declare_shape_loginfo:nn {#1} {#2}
437 }
438 \cs_generate_variant:Nn \@@_declare_shape:nmmm {nxxx}
```

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

`\@@_setup_single_size:nn`

```
439 \cs_new:Nn \@@_setup_single_size:nn
440 {
441 \tl_clear:N \l_@@_size_tl
442 \tl_set_eq:NN \l_@@_sizedfont_tl \l_@@_saved_fontname_tl % in case not spec'ed
443
444 \keys_set_known:nxN {fontspec-sizing} { \exp_after:wN \use:n #2 }
445 \l_@@_sizing_leftover_clist
446 \tl_if_empty:NT \l_@@_size_tl { \@@_error:n {no-size-info} }
447 <debug>\typeout{==~ size:~\l_@@_size_tl}
448
449 % "normal"
450 \@@_load_fontname:Nn \l_fontspeg_fontname_tl {\l_@@_sizedfont_tl}
451 \@@_setup_nfss:Nmmm \l_@@_nfss_tl {#1} {\l_@@_sizing_leftover_clist} {}
452 <debug> \typeout{===~ sized~ font:~ \l_@@_sizedfont_tl}
453
454 % small caps
455 \clist_set_eq:NN \l_@@_fontfeat_curr_clist \l_@@_fontfeat_sc_clist
456
457 \bool_if:NF \l_@@_nosc_bool
458 {
```

```

459     \tl_if_empty:NTF \l_@@_fontname_sc_tl
460     {
461         \@@_make_smallcaps:TF
462         {
463 <debug>\typeout{====~Small~ caps~ found.}
464             \clist_put_left:Nn \l_@@_fontfeat_curr_clist {Letters=SmallCaps}
465             }
466         {
467 <debug>\typeout{====~Small~ caps~ not~ found.}
468             \bool_set_true:N \l_@@_nosc_bool
469             }
470     }
471     { \@@_load_fontname:Nn \l_fontspec_fontname_tl {\l_@@_fontname_sc_tl} }% local for
472 }
473
474 \bool_if:NF \l_@@_nosc_bool
475 {
476     \@@_setup_nfss:Nnnn \l_@@_nfss_sc_tl
477     {#1} {\l_@@_sizing_leftover_clist} {\l_@@_fontfeat_curr_clist}
478 }
479 }

```

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

\@@_setup_nfss:Nnnn

```

480 \cs_new:Nn \@@_setup_nfss:Nnnn
481 {
482 <debug>\typeout{====~Setup-NFSS~shape:~<\l_@@_size_tl>~\l_fontspec_fontname_tl}
483
484     \@@_get_features:n { #2 , #3 , #4 }
485 <debug>\typeout{====~Gathered~features:~\g_@@_rawfeatures_sclist}
486
487     \tl_if_empty:NF \l_@@_scale_tl
488     {
489         \tl_set:Nx \l_@@_scale_tl { s*[\l_@@_scale_tl] }
490     }
491
492     \tl_build_put_right:Nx #1
493     {
494         <\l_@@_size_tl> \l_@@_scale_tl
495         \@@_construct_font_call:nn { \l_fontspec_fontname_tl }
496         { \l_@@_pre_feat_sclist \g_@@_rawfeatures_sclist }
497     }
498 }

```

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

\@@_declare_shapes_normal:nn

```

499 \cs_new:Nn \@@_declare_shapes_normal:nn
500 {
501     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
502     {#1} {#2} {\l_@@_nfss_tl}{\l_@@_postadjust_tl}
503 }

```

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

\@@_declare_shapes_smcaps:nn

```

504 \cs_new:Nn \@@_declare_shapes_smcaps:nn
505 {
506   \tl_if_empty:NF \l_@@_nfss_sc_tl
507   {
508     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl} {#1}
509     { \@@_combo_sc_shape:n {#2} } {\l_@@_nfss_sc_tl} {\l_@@_postadjust_tl}
510   }
511 }
512 \cs_new:Nn \@@_combo_sc_shape:n
513 {
514   \tl_if_exist:cTF { \@@_shape_merge:nn {#1} {\scdefault} }
515   { \tl_use:c { \@@_shape_merge:nn {#1} {\scdefault} } }
516   { \scdefault }
517 }

```

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

\@@_DeclareFontShape:nnnnnn

```

518 \cs_new:Nn \@@_DeclareFontShape:nnnnnn
519 {
520   <debug>\typeout{DeclareFontShape:~{#1}{#2}{#3}{#4}...}
521   \group_begin:
522   \normalsize
523   \cs_undefine:c {#1/#2/#3/#4/\f@size}
524   \group_end:
525   \DeclareFontShape{#1}{#2}{#3}{#4}{#5}{#6}
526   }
527   \cs_generate_variant:Nn \@@_DeclareFontShape:nnnnnn {xxxxxx}

```

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.)

\@@_declare_shape_slanted:nn

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.

```

528 \cs_new:Nn \@@_declare_shape_slanted:nn
529 {
530   \bool_if:nT
531   {
532     \str_if_eq_p:ee {#2} {\itdefault} &&
533     !(\str_if_eq_p:ee {\itdefault} {\sldefault})
534   }
535   {
536     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl}{\g_@@_nfss_family_tl}{#1}{\sldefault}
537     {<->ssub*\g_@@_nfss_family_tl/#1/\itdefault}{\l_@@_postadjust_tl}
538   }
539 }

```


Similar processing for setting up b/bx substitutions.

```

\@@_declare_shapes_bx:nn 540 \cs_new:Nn \@@_declare_shapes_bx:nn
541 {
542   \bool_if:nT
543   {
544     \str_if_eq_p:ee {#1} {\bfdefault} &&
545     !(\str_if_eq_p:ee {\bfdefault} {bx})
546   }
547   {
548     % bx/?
549     \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
550     {bx} {#2}
551     { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
552     { \l_@@_postadjust_tl }
553
554     % bx/sc -> b/sc
555     \tl_if_empty:NF \l_@@_nfss_sc_tl
556     {
557       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
558       {bx} { \@@_combo_sc_shape:n {#2} }
559       { <->ssub*\g_@@_nfss_family_tl/\bfdefault/#2 }
560       { \l_@@_postadjust_tl }
561     }
562
563     % bx/sl -> bx/it
564     \bool_if:nT
565     {
566       \str_if_eq_p:ee {#2} {\itdefault} &&
567       !(\str_if_eq_p:ee {\itdefault} {\sldefault})
568     }
569     {
570       \@@_DeclareFontShape:xxxxxx {\g_@@_nfss_enc_tl} {\g_@@_nfss_family_tl}
571       {bx} {\sldefault}
572       { <->ssub*\g_@@_nfss_family_tl/bx/\itdefault }
573       { \l_@@_postadjust_tl }
574     }
575
576   }
577 }

```

Lastly some informative messaging.

```

\@@_declare_shape_loginfo:nn 578 \cs_new:Nn \@@_declare_shape_loginfo:nn
579 {
580   \tl_gput_right:Nx \g_@@_defined_shapes_tl
581   {
582     \exp_not:n { \ }
583     -- \exp_not:N \str_case:nn {#1/#2}
584     {
585       {\mddefault/\shapedefault} {'normal'~}
586       {\bfdefault/\shapedefault} {'bold'~}
587       {\mddefault/\itdefault} {'italic'~}

```

```

588     {\mddefault/\sldefault} {'slanted'~}
589     {\bfdefault/\itdefault} {'bold~ italic'~}
590     {\bfdefault/\sldefault} {'bold~ slanted'~}
591 } (#1/#2)~
592 with~ NFSS~ spec.:~
593 \l_@@_nfss_tl
594 \exp_not:n { \ }
595 -- \exp_not:N \str_case:nn { #1 / \@@_combo_sc_shape:n {#2} }
596 {
597     {\mddefault/\scdefault} {'small~ caps'~}
598     {\bfdefault/\scdefault} {'bold~ small~ caps'~}
599     {\mddefault/\scitdefault} {'italic~ small~ caps'~}
600     {\bfdefault/\scitdefault} {'bold~ italic~ small~ caps'~}
601     {\mddefault/\scsldefault} {'slanted~ small~ caps'~}
602     {\bfdefault/\scsldefault} {'bold~ slanted~ small~ caps'~}
603 }~( #1 / \@@_combo_sc_shape:n {#2} )~
604 with~ NFSS~ spec.:~
605 \l_@@_nfss_sc_tl
606 \tl_if_empty:FF {\l_@@_postadjust_tl}
607 {
608     \exp_not:N \ and~ font~ adjustment~ code:
609     \exp_not:N \ \l_@@_postadjust_tl
610 }
611 }
612 }

```

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 613 \tl_set:Nn \l_@@_pre_feat_sclist
614 <*XE>
615 {
616     \bool_if:NT \l_@@_ot_bool
617     {
618         \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
619         \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
620     }
621 }
622 </XE>
623 <*LU>
624 {
625     mode = \l_@@_mode_tl ;
626     \tl_if_empty:NF \l_@@_shaper_tl { shaper = \l_@@_shaper_tl ; }
627     \tl_if_empty:NF \l_@@_script_tl { script = \l_@@_script_tl ; }
628     \tl_if_empty:NF \l_@@_lang_tl { language = \l_@@_lang_tl ; }
629 }
630 </LU>

```

This macro checks if the font contains small caps.

```

\@@_make_ot_smallcaps:TF 631 <LU>\cs_new:Nn \@@_make_smallcaps:TF

```

```

632 <XE>\cs_new:Nn \@@_make_ot_smallcaps:TF
633 {
634   \exp_args:No \@@_check_ot_feat:NnTF \l_@@_fontface_cs_tl {smcp} {#1} {#2}
635 }
636 <*XE>
637 \cs_new:Nn \@@_make_smallcaps:TF
638 {
639   \bool_if:NTF \l_@@_ot_bool
640     { \@@_make_ot_smallcaps:TF {#1} {#2} }
641     {
642       \bool_if:NT \l_@@_atsui_bool
643         {
644           \exp_args:No \@@_make_AAT_feature_string:NnnTF
645             \l_@@_fontface_cs_tl {3} {3} {#1} {#2}
646         }
647     }
648 }
649 </XE>

```

\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.

```

\@@_update_featstr:n
650 \cs_new:Nn \@@_update_featstr:n
651 {
652   <debug> \typeout{::: @@_update_featstr:n {#1}}
653   \bool_if:NF \l_@@_firsttime_bool
654   {
655     \tl_gset:Nx \g_@@_single_feat_tl { #1 }
656     <debug> \typeout{:::~ Adding~ feature.}
657     \tl_gput_right:Nx \g_@@_rawfeatures_sclist {#1;}
658   }
659 }

```

```

\@@_remove_clashing_featstr:n
660 \cs_new:Nn \@@_remove_clashing_featstr:n
661 {
662   <debug> \typeout{::: @@_remove_clashing_featstr:n {#1}}
663   \clist_map_inline:nn {#1}
664   {
665     <debug> \typeout{:::~ Removing~ feature~ "##1;}
666     \tl_gremove_all:Nn \g_@@_rawfeatures_sclist {##1;}
667   }
668 }
669 \cs_generate_variant:Nn \@@_remove_clashing_featstr:n {x}

```

1.3 Initialisation

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

```

\@@_init:
670 \cs_set:Npn \@@_init:
671 {

```

```

672 <debug> \typeout{: @@@_init:}
673 \bool_set_false:N \l_@@_ot_bool
674 \bool_set_true:N \l_@@_firsttime_bool
675 \@@_font_is_name:
676 \tl_clear:N \l_@@_font_path_tl
677 \tl_clear:N \l_@@_optical_size_tl
678 \tl_clear:N \l_@@_ttc_index_tl
679 \tl_clear:N \l_@@_renderer_tl
680 \tl_gclear:N \g_@@_defined_shapes_tl
681 \tl_gclear:N \g_@@_curr_series_tl
682 \tl_gset_eq:NN \g_@@_nfss_enc_tl \g_fontspeg_encoding_tl
683 <*LU>
684 \tl_set:Nn \l_@@_mode_tl {node}
685 \int_set:Nn \prehyphenchar { \- } % fixme
686 \int_zero:N \posthyphenchar % fixme
687 \int_zero:N \preexhyphenchar % fixme
688 \int_zero:N \postexhyphenchar % fixme
689 </LU>
690 }

```

Executed in \@@_get_features:Nn.

```

\@@_init_fontface:
691 \cs_new:Nn \@@_init_fontface:
692 {
693 \tl_gclear:N \g_@@_rawfeatures_sclist
694 \tl_clear:N \l_@@_scale_tl
695 \tl_set_eq:NN \l_@@_opacity_tl \c_@@_opacity_tl
696 \tl_set_eq:NN \l_@@_hexcol_tl \c_@@_hexcol_tl
697 \tl_set_eq:NN \l_@@_postadjust_tl \c_@@_postadjust_tl
698 \tl_clear:N \l_@@_wordspace_adjust_tl
699 \tl_clear:N \l_@@_punctspace_adjust_tl
700 }

```

1.4 Miscellaneous

This macro takes an OpenType tag and validates it.

```

\@@_ot_validate_tag:n
701 <*LU>
702 \cs_new_protected:Nn \@@_ot_validate_tag:n
703 {
704 \@@_ot_validate_tag:w #1 \q_nil
705 }
706 \cs_generate_variant:Nn \@@_ot_validate_tag:n {x}
707 \cs_set:Npn \@@_ot_validate_tag:w #1 #2 \q_nil
708 {
709 \bool_if:nTF { \str_if_eq_p:nn {#1} {+} || \str_if_eq_p:nn {#1} {-} }
710 { \@@_ot_validate_tag_aux:w #2 \c_empty_tl \c_empty_tl \q_nil }
711 { \@@_ot_validate_tag_aux:w #1#2 \c_empty_tl \c_empty_tl \q_nil }
712 }
713 \cs_set:Npn \@@_ot_validate_tag_aux:w #1#2#3#4#5 \q_nil
714 {
715 \int_compare:nT { \tl_count:n {#5} > 2 }

```

```

716     { \@@_error:nx {ot-tag-too-long} {#1#2#3#4#5} }
717   }
718 </LU>

```

`\@@_iv_str_to_num:Nn` This macro takes a four character string and converts it to the numerical representation required for X_YTeX 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.

```

719 <*XE>
720 \cs_new:Nn \@@_iv_str_to_num:Nn
721   {
722   <debug>\typeout{iv_str_to_num:~#1~/~#2}
723     \@@_strip_leading_sign:Nw #1#2 \q_nil
724   }
725 \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 -.

```

726 \cs_set:Npn \@@_strip_leading_sign:Nw #1#2#3 \q_nil
727   {
728     \bool_if:nTF { \str_if_eq_p:nn {#2} {+} || \str_if_eq_p:nn {#2} {-} }
729       { \@@_iv_str_to_num:w #1 \q_nil #3 \c_empty_tl \c_empty_tl \q_nil }
730       { \@@_iv_str_to_num:w #1 \q_nil #2#3 \c_empty_tl \c_empty_tl \q_nil }
731   }

```

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

```

732 \cs_set:Npn \@@_iv_str_to_num:w #1 \q_nil #2#3#4#5#6 \q_nil
733   {
734     \int_compare:nT { \tl_count:n {#6} > 2 }
735       { \@@_error:nx {ot-tag-too-long} {#2#3#4#5#6} }
736
737     \int_set:Nn #1
738       {
739         `#2 * "10000000
740         + `#3 * "100000
741         + \ifx \c_empty_tl #4 32 \else `#4 \fi * "100
742         + \ifx \c_empty_tl #5 32 \else `#5 \fi
743       }
744   }
745 </XE>

```

File XI

fontspec-code-opentype.dtx

1 OpenType definitions code

```
\@@_define_opentype_feature_group:n 1 \cs_new:Nn \@@_define_opentype_feature_group:n
2 {
3   \keys_define:nn {fontspec-opentype} { #1 .multichoice: , .groups:n = {opentype} }
4 }

#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)

\@@_define_opentype_feature:nmnnn 5 \cs_new:Nn \@@_feat_prop_add:n
6 {
7   \tl_if_empty:nF {#1}
8   {
9     \prop_if_in:NnF \g_@@_OT_features_prop {#1}
10    {
11      \prop_gput:Nnn \g_@@_OT_features_prop {#1} {#2}
12    }
13  }
14 }
15 \cs_new:Nn \@@_define_opentype_feature:nmnnn
16 {
17   \@@_feat_prop_add:n {#3} {#1\,=#, #2}
18   \tl_if_empty:nTF {#4}
19   {
20     \keys_define:nn {fontspec-opentype}
21     {
22       #1/#2 .code:n =
23       { \@@_remove_clashing_featstr:n {#5} } ,
24       #1/#2 .groups:n = {opentype}
25     }
26   }
27   {
28     \keys_define:nn {fontspec-opentype}
29     {
30       #1/#2 .code:n =
31       {
32         (debug) \typeout{::::::::::fontspec-opentype~#1/#2~=#3/#4/#5}
33         \@@_make_OT_feature:nmn {#3} {#4} {#5}
34       } ,
35       #1/#2 .groups:n = {opentype}
36     }
37   }
```

```

37     }
38 }

#1 : Feature key
\@@_define_opentype_onoffreset:nnnn #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)
39 \cs_new:Nn \@@_feat_off:n {#10ff}
40 \cs_new:Nn \@@_feat_reset:n {#1Reset}
41 \cs_new:Nn \@@_define_opentype_onoffreset:nnnn
42 {
43   \exp_args:Nnx \@@_define_opentype_feature:nnnn {#1} {#2} {#3} {+#4} {#5}
44   \exp_args:Nnx \@@_define_opentype_feature:nnnn {#1} { \@@_feat_off:n {#2} } {#3} {-#4}
45   \exp_args:Nnx \@@_define_opentype_feature:nnnn {#1} { \@@_feat_reset:n {#2} } {} {} {+#4}
46 }

#1 : Feature key
\@@_define_opentype_onreset:nnnn #2 : Feature option val
#3 : Check feature
#4 : Exact tag string to activate
#5 : Tags to remove (clist)
47 \cs_new:Nn \@@_define_opentype_onreset:nnnn
48 {
49   \exp_args:Nnx \@@_define_opentype_feature:nnnn {#1} {#2} {#3} {#4} {#5}
50   \exp_args:Nnx \@@_define_opentype_feature:nnnn {#1} { \@@_feat_reset:n {#2} } {} {} {#4}
51 }

```

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.

```

52 \cs_new:Nn \@@_make_OT_feature:nnn
53 {
54   <debug> \typeout{: @@@_make_OT_feature:nnn \exp_not:n { {#1}{#2}{#3} } }
55
56   \bool_set_true:N \l_@@_proceed_bool
57
58   \tl_if_empty:nF {#1}
59   {
60     \exp_args:No \@@_check_ot_feat:NnF \l_@@_fontface_cs_tl {#1}
61     {
62       \@@_warning:nx {icu-feature-not-exist-in-font} {#1}
63       \bool_set_false:N \l_@@_proceed_bool
64     }

```

```

65     }
66
67     \@@_remove_clashing_featstr:x { #2 , \@@_swap_plus_minus:n {#2} , #3 }
68
69     \bool_if:NT \l_@@_proceed_bool { \@@_update_featstr:n {#2} }
70   }
71   \cs_generate_variant:Nn \@@_make_OT_feature:nnn {xxx}
72   \cs_new:Nn \@@_swap_plus_minus:n { \@@_swap_plus_minus_aux:Nq #1 \q_nil }
73   \cs_new:Npn \@@_swap_plus_minus_aux:Nq #1#2 \q_nil
74   { \str_case:nn {#1} { {+} {-#2} {-} {+#2} } }

```

(End definition for \@@_DeclareFontShape:nnnnn 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.

```

75   \prg_new_conditional:Nnn \@@_check_script:Nn {TF,T}
76   {
77   <debug>\typeout{:: _check_script:Nn~#1~/~#2}
78     \bool_if:NTF \l_@@_never_check_bool
79     { \prg_return_true: }
80     {
81     \bool_if:NTF { \tl_if_empty_p:e {#2} }
82     { \prg_return_false: }
83     {
84     <*XE>
85     <debug>\typeout{:::~ checking~ script~ #2}
86     \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
87     \int_set:Nn \l_tmpb_int { \XeTeXOTcountscript #1 }
88     \int_zero:N \l_tmpa_int
89     \bool_set_false:N \l__fontspec_check_bool
90     \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
91     {
92     \ifnum \XeTeXOTscripttag #1 \l_tmpa_int = \l_@@_strnum_int
93     \bool_set_true:N \l__fontspec_check_bool
94     \int_set:Nn \l_tmpa_int { \l_tmpb_int }
95     \else
96     \int_incr:N \l_tmpa_int
97     \fi
98     }
99     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
100   </XE>
101   <*LU>
102     \@@_ot_validate_tag:x {#2}
103     \cs_if_eq:NNTF #1 \font
104     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
105     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
106   <debug>\typeout{:::~ checking:~"\l_@@_tmp_tl",~ "#2"}
107     \lua_now:e { fontspec.check_ot_script("\l_@@_tmp_tl", "#2") }
108     \bool_if:NTF \l__fontspec_check_bool
109     {
110   <debug>\typeout{:::~ TRUE}

```



```

117         \prg_return_true:
118     }
119     {
120 <debug>\typeout{::::::~~ FALSE}
121         \prg_return_false:
122     }
123 </LU>
124 }
125 }
126 }

```

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

\@@_check_lang:NnnTF 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'.

```

127 \prg_new_conditional:Nnn \@@_check_lang:Nn {TF}
128 {
129     \@@_check_lang:NnnTF #1 {#2} {\l_@@_script_int} {\prg_return_true:} {\prg_return_false:}
130 }
131
132 \prg_new_conditional:Nnn \@@_check_lang:Nnn {TF}
133 {
134 <debug>\typeout{:: _check_lang:Nn~#1~/~#2~/~#3~/}
135     \bool_if:NTF \l_@@_never_check_bool
136     { \prg_return_true: }
137     {
138         \bool_if:NTF { \tl_if_empty_p:e {#3} }
139         { \prg_return_false: }
140     }
141 <*XE>
142     \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
143     \@@_iv_str_to_num:Nx \l_@@_script_int {#3}
144     \int_set:Nn \l_@@_tmpb_int
145     { \XeTeXOTcountlanguages #1 \l_@@_script_int }
146     \int_zero:N \l_@@_tmpa_int
147     \bool_set_false:N \l_@@_fontspec_check_bool
148     \bool_until_do:nn { \int_compare_p:nNn \l_@@_tmpa_int = \l_@@_tmpb_int }
149     {
150         \int_set:Nn \l_@@_tmpc_int
151         { \XeTeXOTlanguagetag #1 \l_@@_script_int \l_@@_tmpa_int }
152
153         \int_compare:nNnTF \l_@@_tmpc_int = \l_@@_strnum_int
154         {
155             \bool_set_true:N \l_@@_fontspec_check_bool
156             \int_set:Nn \l_@@_tmpa_int {\l_@@_tmpb_int}
157         }
158         {
159             \int_incr:N \l_@@_tmpa_int
160         }
161     }
162 }

```

```

155     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
156 </XE>
157 <*LU>
158     \@@_ot_validate_tag:x {#2}
159     \@@_ot_validate_tag:x {#3}
160     \cs_if_eq:NNTF #1 \font
161       { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
162       { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
163     \@@_lua_function:neee {check_ot_lang} {\l_@@_tmp_tl} {#2} {#3}
164     \bool_if:NTF \l__fontspec_check_bool \prg_return_true: \prg_return_false:
165 </LU>
166   }
167 }
168 }

```

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

\@@_check_ot_feat:NnTF This macro takes an OpenType feature tag and checks if it exists in the current font/script/language.
 \@@_check_ot_feat:NnnnTF \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'.

```

169 \prg_new_conditional:Nnn \@@_check_ot_feat:Nn {TF,F}
170 {
171   \@@_check_ot_feat:NnnnTF #1 {#2} {\l_@@_lang_tl} {\l_@@_script_tl}
172   {\prg_return_true:} {\prg_return_false:}
173 }
174 \prg_new_conditional:Nnn \@@_check_ot_feat:Nnnn {TF,F}
175 {
176   \bool_if:NTF \l_@@_never_check_bool
177   { \prg_return_true: }
178   {
179     \bool_if:NTF { \tl_if_empty_p:e {#3} || \tl_if_empty_p:e {#4} }
180     { \prg_return_false: }
181     {
182 <*XE>
183 <debug> \typeout{::~~ fontspec_check_ot_feat:nnn~ {#2}{#3}{#4}}
184     \@@_iv_str_to_num:Nx \l_@@_strnum_int {#2}
185
186     \str_if_eq:eeTF {#3} {dflt}
187     { \int_zero:N \l_@@_language_int }
188     { \@@_iv_str_to_num:Nx \l_@@_language_int {#3} }
189     \@@_iv_str_to_num:Nx \l_@@_script_int {#4}
190
191     \int_set:Nn \l_tmpb_int
192     { \XeTeXOTcountfeatures #1 \l_@@_script_int \l_@@_language_int }
193
194     \int_zero:N \l_tmpa_int
195     \bool_set_false:N \l_@@_check_bool
196     \bool_until_do:nn { \int_compare_p:nNn \l_tmpa_int = \l_tmpb_int }
197     {
198       \ifnum\XeTeXOTfeaturetag #1 \l_@@_script_int \l_@@_language_int

```

```

199         \l_tmpa_int =\l_@@_strnum_int
200         \bool_set_true:N \l_@@_check_bool
201         \int_set:Nn \l_tmpa_int {\l_tmpb_int}
202     \else
203         \int_incr:N \l_tmpa_int
204     \fi
205 }
206 \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
207 </XE>
208 <*LU>
209 <debug>\typeout{::~~ fontspec_check_ot_feat:n~ {#1}}
210     \@@_ot_validate_tag:x {#2}
211     \@@_ot_validate_tag:x {#3}
212     \@@_ot_validate_tag:x {#4}
213     \cs_if_eq:NNTF #1 \font
214     { \tl_set:Nx \l_@@_tmp_tl {\curr@fontshape/\f@size} }
215     { \tl_set:Nx \l_@@_tmp_tl {\cs_to_str:N #1} }
216     \@@_lua_function:neeee {check_ot_feat} {\l_@@_tmp_tl} {#2} {#3} {#4}
217     \bool_if:NTF \l_@@_check_bool \prg_return_true: \prg_return_false:
218 </LU>
219 }
220 }
221 }

```

(End definition for \@@_check_ot_feat:NnTF and \@@_check_ot_feat:NnnnTF. These functions are documented on page ??.)

1.2 OpenType feature information

```

222 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {aalt}{Access-All-Alternates}
223 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvf}{Above-base-Forms}
224 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvm}{Above-base-Mark-Positioning}
225 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {abvs}{Above-base-Substitutions}
226 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {afrc}{Alternative-Fractions}
227 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {akhn}{Akhands}
228 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwf}{Below-base-Forms}
229 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blwm}{Below-base-Mark-Positioning}
230 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {blws}{Below-base-Substitutions}
231 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {calt}{Contextual-Alternates}
232 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {case}{Case-Sensitive-Forms}
233 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ccmp}{Glyph-Composition~/~Decomposition}
234 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cfar}{Conjunct-Form-After-Ro}
235 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cجت}{Conjunct-Forms}
236 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {clig}{Contextual-Ligatures}
237 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpct}{Centered-CJK-Punctuation}
238 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cpsp}{Capital-Spacing}
239 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cswh}{Contextual-Swash}
240 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {curs}{Cursive-Positioning}
241 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {cvNN}{Character-Variant-~N$}
242 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2pc}{Petite-Capitals-From-Capitals}
243 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {c2sc}{Small-Capitals-From-Capitals}
244 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dist}{Distances}

```

245 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dlig}{Discretionary~Ligatures}
246 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dnom}{Denominators}
247 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {dtls}{Dotless~Forms}
248 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {expt}{Expert~Forms}
249 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {falt}{Final~Glyph~on~Line~Alternates}
250 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin2}{Terminal~Forms~\#2}
251 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fin3}{Terminal~Forms~\#3}
252 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fina}{Terminal~Forms}
253 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {flac}{Flattened~accent~forms}
254 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {frac}{Fractions}
255 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {fwid}{Full~Widths}
256 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {half}{Half~Forms}
257 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {haln}{Halant~Forms}
258 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {halt}{Alternate~Half~Widths}
259 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hist}{Historical~Forms}
260 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hkna}{Horizontal~Kana~Alternates}
261 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hlig}{Historical~Ligatures}
262 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hngl}{Hangul}
263 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hojo}{Hojo~Kanji~Forms}
264 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {hwid}{Half~Widths}
265 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {init}{Initial~Forms}
266 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {isol}{Isolated~Forms}
267 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ital}{Italics}
268 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jalt}{Justification~Alternates}
269 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp78}{JIS78~Forms}
270 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp83}{JIS83~Forms}
271 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp90}{JIS90~Forms}
272 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {jp04}{JIS2004~Forms}
273 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {kern}{Kerning}
274 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lfb}{Left~Bounds}
275 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {liga}{Standard~Ligatures}
276 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ljmo}{Leading~Jamo~Forms}
277 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {lnum}{Lining~Figures}
278 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {locl}{Localized~Forms}
279 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltra}{Left-to-right~alternates}
280 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ltrm}{Left-to-right~mirrored~forms}
281 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mark}{Mark~Positioning}
282 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {med2}{Medial~Forms~\#2}
283 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {medi}{Medial~Forms}
284 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mgrk}{Mathematical~Greek}
285 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mkmk}{Mark~to~Mark~Positioning}
286 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {mset}{Mark~Positioning~via~Substitution}
287 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nalt}{Alternate~Annotation~Forms}
288 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nlck}{NLC~Kanji~Forms}
289 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {nukt}{Nukta~Forms}
290 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {numr}{Numerators}
291 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {onum}{Oldstyle~Figures}
292 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {opbd}{Optical~Bounds}
293 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ordn}{Ordinals}
294 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ornm}{Ornaments}
295 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {palt}{Proportional~Alternate~Widths}

296 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pcap}{Petite-Capitals}
297 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pkna}{Proportional-Kana}
298 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pnum}{Proportional-Figures}
299 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pref}{Pre-Base-Forms}
300 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pres}{Pre-base-Substitutions}
301 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pstf}{Post-base-Forms}
302 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {psts}{Post-base-Substitutions}
303 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {pwid}{Proportional-Widths}
304 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {qwid}{Quarter-Widths}
305 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rand}{Randomize}
306 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rclt}{Required-Contextual-Alternates}
307 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rkrf}{Rakar-Forms}
308 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rlig}{Required-Ligatures}
309 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rphf}{Reph-Forms}
310 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtbd}{Right-Bounds}
311 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtla}{Right-to-left-alternates}
312 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rtlm}{Right-to-left-mirrored-forms}
313 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ruby}{Ruby-Notation-Forms}
314 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {rvrn}{Required-Variation-Alternates}
315 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {salt}{Stylistic-Alternates}
316 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sinf}{Scientific-Inferiors}
317 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {size}{Optical-size}
318 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smcp}{Small-Capitals}
319 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {smpl}{Simplified-Forms}
320 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssNN}{Stylistic-Set- $\$N\$}$
321 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {ssty}{Math-script-style-alternates}
322 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {stch}{Stretching-Glyph-Decomposition}
323 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {subs}{Subscript}
324 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {sup}{Superscript}
325 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {swsh}{Swash}
326 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {titl}{Titling}
327 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tjmo}{Trailing-Jamo-Forms}
328 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnam}{Traditional-Name-Forms}
329 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {tnum}{Tabular-Figures}
330 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {trad}{Traditional-Forms}
331 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {twid}{Third-Widths}
332 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {unic}{Unicase}
333 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {valt}{Alternate-Vertical-Metrics}
334 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vatu}{Vattu-Variants}
335 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vert}{Vertical-Writing}
336 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vhal}{Alternate-Vertical-Half-Metrics}
337 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vjmo}{Vowel-Jamo-Forms}
338 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkna}{Vertical-Kana-Alternates}
339 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vkrn}{Vertical-Kerning}
340 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vpal}{Proportional-Alternate-Vertical-Metrics}
341 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrt2}{Vertical-Alternates-and-Rotation}
342 \prop_gput:Nnn \g_@@_all_opentype_feature_names_prop {vrtr}{Vertical-Alternates-for-Rotation}
343 \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`

```
1 \cs_new:Nn \@@_define_aat_feature_group:n
2 {
3   \keys_define:nn {fontspec-aat} { #1 .multichoice: }
4 }
```

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

`\@@_define_aat_feature:nmmn`

```
5 \cs_new:Nn \@@_define_aat_feature:nmmn
6 {
7   \keys_define:nn {fontspec-aat}
8     {
9       #1/#2 .code:n = { \@@_make_AAT_feature:nn {#3}{#4} }
10    }
11 }
```

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

`\@@_make_AAT_feature:nn`

```
12 \cs_new:Nn \@@_make_AAT_feature:nn
13 {
14   \tl_if_empty:nTF {#1}
15     { \@@_warning:n {aat-feature-not-exist} }
16     {
17       \exp_args:No \@@_make_AAT_feature_string:NmmTF \l_@@_fontface_cs_tl {#1} {#2}
18         {
19           \@@_update_featstr:n {\l_fontspec_feature_string_tl}
20         }
21         {
22           \@@_warning:nx {aat-feature-not-exist-in-font} {#1,#2}
23         }
24       }
25 }
```

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

`\@@_make_AAT_feature_string:NmmTF`

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 58).

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 X_YL_AT_EX 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_fontspeg_feature_string_tl`.

```

26 \prg_new_conditional:Nnn \@@_make_AAT_feature_string:Nnn {TF,T,F}
27 {
28   \tl_set:Nx \l_@@_tmpa_tl { \XeTeXfeaturename #1 #2 }
29   \tl_if_empty:NTF \l_@@_tmpa_tl
30     { \prg_return_false: }
31     {
32       \int_compare:nTF { \XeTeXisexclusivefeature #1 #2 > 0 }
33         {
34           \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2\space #3}
35         }
36         {
37           \int_if_even:nTF {#3}
38             {
39               \tl_set:Nx \l_@@_tmpb_tl { \XeTeXselectorname #1 #2\space #3}
40             }
41             {
42               \tl_set:Nx \l_@@_tmpb_tl
43                 {
44                   \XeTeXselectorname #1 #2\space \numexpr#3-1\relax
45                 }
46               \tl_if_empty:NF \l_@@_tmpb_tl { \tl_put_left:Nn \l_@@_tmpb_tl {!} }
47             }
48         }
49
50       \tl_if_empty:NTF \l_@@_tmpb_tl
51         { \prg_return_false: }
52         {
53           \tl_set:Nx \l_fontspeg_feature_string_tl { \l_@@_tmpa_tl = \l_@@_tmpb_tl }
54           \prg_return_true:
55         }
56     }
57 }

```

(End 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.

```
1 \clist_gset:Nn \g_@@_all_keyval_modules_clist
2 {
3   fontspec, fontspec-opentype, fontspec-aat,
4   fontspec-preparse, fontspec-preparse-cfg, fontspec-preparse-external, fontspec-preparse-n
5   fontspec-renderer
6 }
```

Wrapper function to save some characters in the source:

```
7 \cs_new:Nn \@@_keys_define_code:nnn
8 {
9   \keys_define:nn {#1} { #2 .code:n = {#3} }
10 }
```

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

```
11 \cs_new:Nn \@@_aff_error:n
12 {
13   \@@_keys_define_code:nnn {fontspec-addfeatures} {#1}
14   { \@@_error:nx {not-in-addfontfeatures} {#1} }
15 }
```

1.1 Pre-pre-parsing stages

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

Don't load font config file

```
16 \@@_keys_define_code:nnn {fontspec-preparse-cfg} {IgnoreFontspecFile}
17 {
18   \bool_set_false:N \l_@@_fontcfg_bool
19 }
20 \@@_keys_define_code:nnn {fontspec-preparse-external} {IgnoreFontspecFile}
21 {
22   \bool_set_false:N \l_@@_fontcfg_bool
23 }
```

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.

```
24 \@@_keys_define_code:nnn {fontspec-preparse-external} {Path}
25 {
26   \bool_set_true:N \l_@@_nobf_bool
27   \bool_set_true:N \l_@@_noit_bool
28   \bool_set_true:N \l_@@_external_bool
```



```

29     \tl_set:Nn \l_@@_font_path_tl {#1}
30     \@@_font_is_file:
31 <*XE>
32     \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
33 </XE>
34   }
35 \aliasfontfeature{Path}{ExternalLocation}
36 \@@_keys_define_code:nnn {fontspec} {Path} {}

```

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

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

```

37 \@@_keys_define_code:nnn {fontspec-prepare-external} {Extension}
38   {
39     \tl_set:Nn \l_@@_extension_tl {#1}
40     \bool_if:NF \l_@@_external_bool
41       {
42         \keys_set:nn {fontspec-prepare-external} {Path}
43       }
44   }
45 \tl_clear:N \l_@@_extension_tl
46 \@@_keys_define_code:nnn {fontspec} {Extension} {}

```

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.

```

47 <*XE>
48 \keys_define:nn {fontspec-renderer}
49   {
50     Renderer .choices:nn =
51       {AAT,ICU,OpenType,Graphite,Full,Basic,Node,Base,HarfBuzz,Harfbuzz}
52     {
53       \int_compare:nTF {\l_keys_choice_int <= 4}
54         {
55           \tl_set:Nx \l_@@_renderer_tl
56             {
57               \int_case:nn \l_keys_choice_int { 1 {/AAT} 2 {/OT} 3 {/OT} 4 {/GR} }
58             }
59 <debug>\typeout{Renderer:~ \l_@@_renderer_tl}
60           \tl_gset:Nx \g_@@_single_feat_tl { \l_@@_renderer_tl }
61         }
62       {
63         \@@_warning:nx {only-luatex-feature} {Renderer=Full/Basic/Node/Base/HarfBuzz}
64       }
65     }
66   }
67 </XE>
68 <*LU>
69 \keys_define:nn {fontspec-renderer}
70   {
71     Renderer .choices:nn =

```

```

72 {Full,Node,Basic,Base,HarfBuzz,Harfbuzz,OpenType,AAT,Graphite}
73 {
74   \int_compare:nT {\l_keys_choice_int >= 5} { \bool_set_true:N \l_@@_harfbuzz_bool }
75
76   \tl_set:Nx \l_@@_mode_tl
77   {
78     \int_case:nn \l_keys_choice_int { 1 {node} 2 {node} 3 {base} 4 {base} 5 {harf} 6
79     }
80
81   \tl_set:Nx \l_@@_shaper_tl
82   {
83     \int_case:nn \l_keys_choice_int { 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {ot} 8 {coretex
84     }
85
86   <debug> \typeout{Mode:~"\l_@@_mode_tl"~/~Shaper:~"\l_@@_shaper_tl"}
87
88   \tl_gset:Nx \g_@@_single_feat_tl
89   {
90     mode=\l_@@_mode_tl ;
91     \tl_if_empty:NF \l_@@_shaper_tl { shaper=\l_@@_shaper_tl}
92   }
93   } ,
94
95   Renderer unknown .code:n =
96   {
97     \bool_set_true:N \l_@@_harfbuzz_bool
98     @@_warning:nx {unknown-renderer} {#1}
99     \tl_set:Nn \l_@@_mode_tl {harf}
100    \tl_set:Nn \l_@@_shaper_tl {#1}
101  } ,
102 }
103 </LU>

```

1.2 Pre-parsed features

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

```

104 \@@_keys_define_code:nnn {fontspec-prepare} {Script}
105 {
106 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
107 \tl_set:Nn \l_@@_script_name_tl {#1}
108 }

```

Exactly the same:

```

109 \@@_keys_define_code:nnn {fontspec-prepare} {Language}
110 {
111 <XE> \keys_set:nn {fontspec-renderer} {Renderer=OpenType}
112 \tl_set:Nn \l_@@_lang_name_tl {#1}
113 }

```

TTC font index

```
114 \@@_keys_define_code:nmn {fontspec-prepare} {FontIndex}
115 {
116   \str_if_eq:eeF { \str_lowercase:f {\l_@@_extension_tl} } {.ttc}
117   { \@@_warning:n {font-index-needs-ttc} }
118   <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
119   <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
120 }
121 \@@_keys_define_code:nmn {fontspec} {FontIndex}
122 {
123   <XE> \tl_set:Nn \l_@@_ttc_index_tl {:#1}
124   <LU> \tl_set:Nn \l_@@_ttc_index_tl {(#1)}
125 }
```

1.3 Font faces

Upright

```
126 \@@_keys_define_code:nmn {fontspec-prepare-external} {UprightFont}
127 {
128   \fontspec_complete_fontname:Nn \l_@@_fontname_up_tl {#1}
129 }
```

Italic and slanted

```
130 \@@_keys_define_code:nmn {fontspec-prepare-external} {ItalicFont}
131 {
132   \tl_if_empty:nTF {#1}
133   {
134     \bool_set_true:N \l_@@_noit_bool
135   }
136   {
137     \bool_set_false:N \l_@@_noit_bool
138     \fontspec_complete_fontname:Nn \l_@@_fontname_it_tl {#1}
139   }
140 }
141 \@@_keys_define_code:nmn {fontspec-prepare-external} {SlantedFont}
142 {
143   \fontspec_complete_fontname:Nn \l_@@_fontname_sl_tl {#1}
144 }
```

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!

```
145 %\@@_keys_define_code:nmn {fontspec-prepare-external} {BoldSeries}
146 % {
147 %   \tl_gset:Nx \g_@@_curr_series_tl { #1 }
148 %   \seq_put_right:Nx \l_@@_bf_series_seq { #1 }
149 % }
```

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

```
150 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldFont}
151 {
152   \tl_if_empty:nTF {#1}
153   {
154     \bool_set_true:N \l_@@_nobf_bool
155   }
156   {
157     \bool_set_false:N \l_@@_nobf_bool
158     \fontspec_complete_fontname:Nn \l_@@_curr_bfname_tl {#1}
159
160     \seq_if_empty:NT \l_@@_bf_series_seq
161     {
162       \tl_gset:Nx \g_@@_curr_series_tl {\bfdefault}
163       \seq_put_right:Nx \l_@@_bf_series_seq {\bfdefault}
164     }
165
166     \tl_if_eq:oxT \g_@@_curr_series_tl {\bfdefault}
167     {
168       \tl_set_eq:NN \l_@@_fontname_bf_tl \l_@@_curr_bfname_tl
169     }
170
171     \prop_put:NxV \l_@@_nfss_prop {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
172
173     <debug>\typeout{Setting~bold~font~"\l_@@_curr_bfname_tl"~with~series~"\g_@@_curr_series_tl"}
174
175   }
176 }
```

Bold italic/slanted

```
177 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldItalicFont}
178 {
179   \fontspec_complete_fontname:Nn \l_@@_fontname_bfit_tl {#1}
180 }
181 \@@_keys_define_code:nnn {fontspec-prepare-external} {BoldSlantedFont}
182 {
183   \fontspec_complete_fontname:Nn \l_@@_fontname_bfsl_tl {#1}
184 }
```

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

```
185 \@@_keys_define_code:nnn {fontspec} {SmallCapsFont}
186 {
187   \tl_if_empty:nTF {#1}
188   {
189     \bool_set_true:N \l_@@_nosc_bool
190   }
191   {
192     \bool_set_false:N \l_@@_nosc_bool
193     \fontspec_complete_fontname:Nn \l_@@_fontname_sc_tl {#1}
194   }
195 }
```

```
195 }
```

1.3.1 Prepared font features

```
196 \@@_keys_define_code:nmn {fontspec-prepare} {UprightFeatures}
197 {
198   \clist_put_right:Nn \l_@@_fontfeat_up_clist {#1}
199 }
200 \@@_keys_define_code:nmn {fontspec-prepare} {BoldFeatures}
201 {
202   \clist_put_right:Nn \l_@@_fontfeat_bf_clist {#1}
203 }
204 % \prop_put:NxV \l_@@_nfss_prop
205 %   {BoldFont-\g_@@_curr_series_tl} \l_@@_curr_bfname_tl
206 }
207 \@@_keys_define_code:nmn {fontspec-prepare} {ItalicFeatures}
208 {
209   \clist_put_right:Nn \l_@@_fontfeat_it_clist {#1}
210 }
211 \@@_keys_define_code:nmn {fontspec-prepare} {BoldItalicFeatures}
212 {
213   \clist_put_right:Nn \l_@@_fontfeat_bfit_clist {#1}
214 }
215 \@@_keys_define_code:nmn {fontspec-prepare} {SlantedFeatures}
216 {
217   \clist_put_right:Nn \l_@@_fontfeat_sl_clist {#1}
218 }
219 \@@_keys_define_code:nmn {fontspec-prepare} {BoldSlantedFeatures}
220 {
221   \clist_put_right:Nn \l_@@_fontfeat_bfsl_clist {#1}
222 }
```

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

```
223 \@@_keys_define_code:nmn {fontspec} {SmallCapsFeatures}
224 {
225   \bool_if:NF \l_@@_firsttime_bool
226   {
227     \clist_put_right:Nn \l_@@_fontfeat_sc_clist {#1}
228   }
229 }
```

Features varying by size

```
230 \@@_keys_define_code:nmn {fontspec-prepare} {SizeFeatures}
231 {
232   \clist_set:Nn \l_@@_sizefeat_clist {#1}
233   \clist_put_right:Nn \l_@@_fontfeat_up_clist { SizeFeatures = {#1} }
234 }
235 \@@_keys_define_code:nmn {fontspec-prepare-nested} {SizeFeatures}
236 {
237   \clist_set:Nn \l_@@_sizefeat_clist {#1}
238   \tl_if_empty:NT \l_@@_this_font_tl
239   { \tl_set:Nn \l_@@_this_font_tl { -- } } % needs to be non-empty as a flag
240 }
```

```

241 \@@_keys_define_code:nmn {fontspec-prepare-nested} {Font}
242 {
243   \tl_set:Nn \l_@@_this_font_tl {#1}
244 }
245 \@@_keys_define_code:nmn {fontspec} {SizeFeatures}
246 {
247   % dummy
248 }
249 \@@_keys_define_code:nmn {fontspec} {Font}
250 {
251   % dummy
252 }
253 \@@_keys_define_code:nmn {fontspec-sizing} {Size}
254 {
255   \tl_set:Nn \l_@@_size_tl {#1}
256 }
257 \@@_keys_define_code:nmn {fontspec-sizing} {Font}
258 {
259   \fontspec_complete_fontname:Nn \l_@@_sizedfont_tl {#1}
260 }

```

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

```

261 \@@_keys_define_code:nmn {fontspec-opentype} {UprightFont} {}
262 \@@_keys_define_code:nmn {fontspec-opentype} {ItalicFont} {}
263 \@@_keys_define_code:nmn {fontspec-opentype} {SlantedFont} {}
264 \@@_keys_define_code:nmn {fontspec-opentype} {BoldFont} {}
265 \@@_keys_define_code:nmn {fontspec-opentype} {BoldItalicFont} {}
266 \@@_keys_define_code:nmn {fontspec-opentype} {BoldSlantedFont} {}

```

1.4 General font-independent features

These features can be applied to any font.

NFSS encoding For the very brave.

```

267 \@@_keys_define_code:nmn {fontspec-prepare} {NFSSEncoding}
268 {
269   \tl_gset:Nx \g_@@_nfss_enc_tl { #1 }
270 }

```

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.)

```

271 \@@_keys_define_code:nmn {fontspec-prepare} {NFSSFamily}
272 {
273   \tl_set:Nx \l_@@_nfss_fam_tl { #1 }
274 }

```

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

```

275 \@@_keys_define_code:nmn {fontspec-preparse} {FontFace}
276 {
277   \tl_clear:N \l_@@_this_font_tl
278   \clist_set:No \l_@@_arg_clist { \use_iii:nmn #1 }
279   \clist_set_eq:NN \l_@@_this_feat_clist \l_@@_arg_clist
280   \int_compare:nT { \clist_count:N \l_@@_arg_clist = 1 }
281   {
282     <debug>\typeout{FontFace~ parsing:~ one~ clist~ item}
283     \tl_if_in:NnF \l_@@_arg_clist {=}
284     {
285       <debug>\typeout{FontFace~ parsing:~ no~ equals~ =>~ font~ name~ only}
286       \tl_set_eq:NN \l_@@_this_font_tl \l_@@_arg_clist
287       \tl_clear:N \l_@@_this_feat_clist
288     }
289   }
290
291   \@@_add_nfssfont:nmnn
292   { \use_i:nmn #1 } { \use_ii:nmn #1 } { \l_@@_this_font_tl } { \l_@@_this_feat_clist }
293 }

```

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

```

294 \@@_keys_define_code:nmn {fontspec} {Scale}
295 {
296   \str_case:nnF {#1}
297   {
298     {MatchLowercase} { \@@_calc_scale:n {5} }
299     {MatchUppercase} { \@@_calc_scale:n {8} }
300   }
301   { \tl_set:Nx \l_@@_scale_tl {#1} }
302 }

```

ScaleAgain

```

303 \@@_keys_define_code:nmn {fontspec} {ScaleAgain}
304 {
305   \tl_if_empty:NT \l_@@_scale_tl { \tl_set:Nn \l_@@_scale_tl {1} }
306   \tl_set:Nx \l_@@_scale_tl { \fp_eval:n { #1 * \l_@@_scale_tl } }
307   \@@_info:n {set-scale}
308 }

```

`\@@_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 X_YTeX).

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.)

```

309 \cs_new:Nn \@@_calc_scale:n
310 {
311   \group_begin:
312
313   \fontencoding { \encodingdefault }
314   \fontfamily { \familydefault }
315   \selectfont
316
317   \@@_set_font_dimen:NnN \l_@@_tmpa_dim {#1} \font
318   \@@_set_font_dimen:NnN \l_@@_tmpb_dim {#1} \l_@@_fontface_cs_tl
319
320   \tl_set:Nx \l_@@_scale_tl
321   {
322     \fp_eval:n { \dim_to_fp:n {\l_@@_tmpa_dim} /
323                 \dim_to_fp:n {\l_@@_tmpb_dim} }
324   }
325
326   \@@_info:n {set-scale}
327   \exp_args:NNNx
328   \group_end:
329   \tl_set:Nx \l_@@_scale_tl { \l_@@_scale_tl }
330 }

```

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

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

```

331 \cs_new:Nn \@@_set_font_dimen:NnN
332 {
333   \dim_set:Nn #1 { \fontdimen #2 #3 }
334   \dim_compare:nNnT #1 = {0pt}
335   {
336     \settoheight #1
337     {
338       \str_if_eq:nnTF {#3} {\font} \rmfamily #3
339       \int_case:nnF #2
340       {
341         {5} {x} % x-height
342         {8} {X} % cap-height
343       } {?} % "else" clause; never reached.
344     }
345   }
346 }

```

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

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

```

347 \@@_keys_define_code:nmn {fontspec} {WordSpace}
348 {
349   \bool_if:NF \l_@@_firsttime_bool
350   { \fontspec_parse_wordspace:w #1,,,\q_stop }
351 }
352 \@@_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}`.

```

353 \cs_set:Npn \fontspec_parse_wordspace:w #1,#2,#3,#4 \q_stop
354 {
355   \tl_if_empty:nTF {#4}
356   {
357     \tl_set:Nn \l_@@_wordspace_adjust_tl
358     {
359       \fontdimen 2 \font = #1 \fontdimen 2 \font
360       \fontdimen 3 \font = #1 \fontdimen 3 \font
361       \fontdimen 4 \font = #1 \fontdimen 4 \font
362     }
363   }
364   {
365     \tl_set:Nn \l_@@_wordspace_adjust_tl
366     {
367       \fontdimen 2 \font = #1 \fontdimen 2 \font
368       \fontdimen 3 \font = #2 \fontdimen 3 \font
369       \fontdimen 4 \font = #3 \fontdimen 4 \font
370     }
371   }
372 }

```

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

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

```

373 \@@_keys_define_code:nmn {fontspec} {PunctuationSpace}
374 {
375   \str_case_e:nnF {#1}
376   {
377     {WordSpace}
378     {
379       \tl_set:Nn \l_@@_punctspace_adjust_tl
380       { \fontdimen 7 \font = 0 \fontdimen 2 \font }
381     }
382     {TwiceWordSpace}
383     {
384       \tl_set:Nn \l_@@_punctspace_adjust_tl
385       { \fontdimen 7 \font = 1 \fontdimen 2 \font }
386     }
387   }
388   {
389     \tl_set:Nn \l_@@_punctspace_adjust_tl

```

```

390         { \fontdimen 7 \font = #1 \fontdimen 7 \font }
391     }
392 }
393 \@@_aff_error:n {PunctuationSpace}

```

Secret hook into the font-adjustment code

```

394 \@@_keys_define_code:nnn {fontspec} {FontAdjustment}
395 {
396     \tl_put_right:Nx \l_@@_postadjust_tl {#1}
397 }

```

Letterspacing

```

398 \@@_keys_define_code:nnn {fontspec} {LetterSpace}
399 {
400     \@@_update_featstr:n {letterspace=#1}
401 }

```

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.

```

402 \@@_keys_define_code:nnn {fontspec} {HyphenChar}
403 {
404     \str_if_eq:nnTF {#1} {None}
405     {
406         \tl_put_right:Nn \l_@@_postadjust_tl
407         { \@@_primitive_font_set_hyphenchar:Nn \font {-1} }
408     }
409     {
410     <LU>     \@@_warning:nx {only-xetex-feature} {HyphenChar}
411
412         \tl_if_single:nTF {#1}
413         { \tl_set:Nn \l_@@_hyphenchar_tl {`#1} }
414         { \tl_set:Nn \l_@@_hyphenchar_tl { #1} }
415
416         \exp_args:No \@@_primitive_font_glyph_if_exist:NnTF \l_@@_fontface_cs_tl {\l_@@_hyphenchar_tl}
417         {
418             \tl_put_right:Nn \l_@@_postadjust_tl
419             { \@@_primitive_font_set_hyphenchar:Nn \font { \l_@@_hyphenchar_tl } }
420         }
421         { \@@_error:nxx {no-glyph}{\l_fontspeg_fontname_tl}{#1} }
422     }
423 }
424 }
425 \@@_aff_error:n {HyphenChar}

```

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

```

426 \@@_keys_define_code:nnn {fontspec} {Color}
427 {

```

```

428 \cs_if_exist:cTF { \token_to_str:N \color@ #1 }
429 {
430   \convertcolorspec{named}{#1}{HTML}\l_@@_hexcol_tl
431 }
432 {
433   \int_compare:nTF { \tl_count:n {#1} == 6 }
434   { \tl_set:Nn \l_@@_hexcol_tl {#1} }
435   {
436     \int_compare:nTF { \tl_count:n {#1} == 8 }
437     { \fontspec_parse_colour:viii #1 }
438     {
439       \bool_if:NF \l_@@_firsttime_bool
440       { \@@_warning:nx {bad-colour} {#1} }
441     }
442   }
443 }
444 }
445 \cs_set:Npn \fontspec_parse_colour:viii #1#2#3#4#5#6#7#8
446 {
447   \tl_set:Nn \l_@@_hexcol_tl {#1#2#3#4#5#6}
448   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
449   {
450     \bool_if:NF \l_@@_firsttime_bool
451     { \@@_warning:nx {opa-twice-col} {#7#8} }
452   }
453   \tl_set:Nn \l_@@_opacity_tl {#7#8}
454 }
455 \aliasfontfeature{Color}{Colour}
456 \@@_keys_define_code:nnn {fontspec} {Opacity}
457 {
458   \int_set:Nn \l_@@_tmp_int {255}
459   \@@_int_mult_truncate:Nn \l_@@_tmp_int { #1 }
460   \tl_if_eq:NNF \l_@@_opacity_tl \c_@@_opacity_tl
461   {
462     \bool_if:NF \l_@@_firsttime_bool
463     { \@@_warning:nx {opa-twice} {#1} }
464   }
465   \tl_set:Nx \l_@@_opacity_tl
466   {
467     \int_compare:nT { \l_@@_tmp_int <= "F } {0} % zero pad
468     \int_to_hex:n { \l_@@_tmp_int }
469   }
470 }

```

Mapping

```

471 <*XE>
472 \@@_keys_define_code:nnn {fontspec-aat} {Mapping}
473 {
474   \tl_set:Nn \l_@@_mapping_tl { #1 }
475 }

```

```

476 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
477 {
478   \tl_set:Nn \l_@@_mapping_tl { #1 }
479 }
480 </XE>
481 <*LU>
482 \@@_keys_define_code:nmn {fontspec-opentype} {Mapping}
483 {
484   \str_if_eq:nnTF {#1} {tex-text}
485   {
486     \@@_warning:n {no-mapping-ligtext}
487     \msg_redirect_name:nmn {fontspec} {no-mapping-ligtext} {none}
488     \keys_set:nn {fontspec-opentype} { Ligatures=TeX }
489   }
490   { \@@_warning:n {no-mapping} }
491 }
492 </LU>

```

1.4.1 Continuous font axes

```

493 \@@_keys_define_code:nmn {fontspec} {Weight}
494 {
495   \@@_update_featstr:n{weight=#1}
496 }
497 \@@_keys_define_code:nmn {fontspec} {Width}
498 {
499   \@@_update_featstr:n{width=#1}
500 }
501 \@@_keys_define_code:nmn {fontspec} {OpticalSize}
502 <*XE>
503 {
504   \bool_if:NTF \l_@@_ot_bool
505   {
506     \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
507   }
508   {
509     \bool_if:NT \l_@@_mm_bool
510     {
511       \@@_update_featstr:n { optical size = #1 }
512     }
513   }
514   \bool_if:nT { !\l_@@_ot_bool && !\l_@@_mm_bool }
515   {
516     \bool_if:NT \l_@@_firsttime_bool
517     { \@@_warning:nx {no-opticals} {\l_fontspeg_fontname_tl} }
518   }
519 }
520 </XE>
521 <*LU>
522 {
523   \tl_set:Nn \l_@@_optical_size_tl {/ S = #1}
524 }

```

525 </LU>

1.4.2 Font transformations

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

```
526 \keys_define:nn {fontspec}
527   {
528     FakeSlant .code:n =
529       {
530         \@@_update_featstr:n {slant=#1}
531       },
532     FakeSlant .default:n = {0.2}
533   }
534 \keys_define:nn {fontspec}
535   {
536     FakeStretch .code:n =
537       {
538         \@@_update_featstr:n {extend=#1}
539       },
540     FakeStretch .default:n = {1.2}
541   }
542 \keys_define:nn {fontspec}
543   {
544     FakeBold .code:n =
545       {
546         \@@_update_featstr:n {embolden=#1}
547       },
548     FakeBold .default:n = {1.5}
549   }
```

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.)

```
550 \keys_define:nn {fontspec}
551   {
552     AutoFakeSlant .code:n =
553       {
554         \bool_if:NT \l_@@_firsttime_bool
555           {
556             \tl_set:Nn \l_@@_fake_slant_tl {#1}
557             \clist_put_right:Nn \l_@@_fontfeat_it_clist {FakeSlant=#1}
558             \tl_set_eq:NN \l_@@_fontname_it_tl \l_fontspec_fontname_tl
559             \bool_set_false:N \l_@@_noit_bool
560
561             \tl_if_empty:NF \l_@@_fake_embolden_tl
562               {
563                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
564                   {FakeBold=\l_@@_fake_embolden_tl}
```

```

565         \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeSlant=#1}
566         \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
567     }
568 }
569 },
570 AutoFakeSlant .default:n = {0.2}
571 }

```

Same but reversed:

```

572 \keys_define:nn {fontspec}
573 {
574     AutoFakeBold .code:n =
575     {
576         \bool_if:NT \l_@@_firsttime_bool
577         {
578             \tl_set:Nn \l_@@_fake_embolden_tl {#1}
579             \clist_put_right:Nn \l_@@_fontfeat_bf_clist {FakeBold=#1}
580             \tl_set_eq:NN \l_@@_fontname_bf_tl \l_fontspec_fontname_tl
581             \bool_set_false:N \l_@@_nobf_bool
582
583             \tl_if_empty:NF \l_@@_fake_slant_tl
584             {
585                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist
586                 {FakeSlant=\l_@@_fake_slant_tl}
587                 \clist_put_right:Nx \l_@@_fontfeat_bfit_clist {FakeBold=#1}
588                 \tl_set_eq:NN \l_@@_fontname_bfit_tl \l_fontspec_fontname_tl
589             }
590         }
591     },
592     AutoFakeBold .default:n = {1.5}
593 }

```

1.4.3 Raw feature string

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

```

594 \@@_keys_define_code:nnn {fontspec-opentype} {RawFeature}
595 {
596     \@@_update_featstr:n {#1}
597 }
598 \@@_keys_define_code:nnn {fontspec-aat} {RawFeature}
599 {
600     \@@_update_featstr:n {#1}
601 }

```

File XIV

fontspec-code-feat-opentype.dtx

1 OpenType feature definitions

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

2 Regular key=val / tag definitions

2.1 Ligatures

```
6 \@@_define_opentype_feature_group:n {Ligatures}
7 \@@_define_opentype_feature:nnnnn {Ligatures} {ResetAll} {} {}
8 {
9   +dlig,-dlig,+rliq,-rliq,+liga,-liga,+dlig,-dlig,+clig,-clig,+hlig,-hlig,
10  <XE> mapping = tex-text
11  <LU> +tlig,-tlig
12  }
13 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Required} {rliq} {rliq} {}
14 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Common} {liga} {liga} {}
15 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Rare} {dlig} {dlig} {}
16 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Discretionary} {dlig} {dlig} {}
17 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Contextual} {clig} {clig} {}
18 \@@_define_opentype_onoffreset:nnnnn {Ligatures} {Historic} {hlig} {hlig} {}
```

Emulate CM extra ligatures.

```
19 <*XE>
20 \keys_define:nn {fontspec-opentype}
21 {
22   Ligatures / TeX .code:n = { \tl_set:Nn \l_@@_mapping_tl {tex-text} },
23   Ligatures / TeXOff .code:n = { \tl_clear:N \l_@@_mapping_tl },
24   Ligatures / TeXReset .code:n = { \tl_clear:N \l_@@_mapping_tl },
25 }
26 </XE>
27 <LU>\@@_define_opentype_onoffreset:nnnnn {Ligatures} {TeX} {} {tlig} {}
```

2.2 Letters

```
28 \@@_define_opentype_feature_group:n {Letters}
29 \@@_define_opentype_feature:nnnnn {Letters} {ResetAll} {} {}
30 {
31   +case,+smcp,+pcap,+c2sc,+c2pc,+unic,+rand,
32   -case,-smcp,-pcap,-c2sc,-c2pc,-unic,-rand
33 }
34 \@@_define_opentype_onoffreset:nnnnn {Letters} {Uppercase} {case} {case} {}
35 \@@_define_opentype_onoffreset:nnnnn {Letters} {SmallCaps} {smcp} {smcp} {+pcap,+unic}
```

```

36 \@@_define_opentype_onoffreset:nnnnn {Letters} {PetiteCaps} {pcap} {pcap} {+smcp,+unic}
37 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercaseSmallCaps} {c2sc} {c2sc} {+c2pc,+uni}
38 \@@_define_opentype_onoffreset:nnnnn {Letters} {UppercasePetiteCaps} {c2pc} {c2pc} {+c2sc,+uni}
39 \@@_define_opentype_onoffreset:nnnnn {Letters} {Unicase} {unic} {unic} {}
40 \@@_define_opentype_onoffreset:nnnnn {Letters} {Random} {rand} {rand} {}

```

2.3 Numbers

```

41 \@@_define_opentype_feature_group:n {Numbers}
42 \@@_define_opentype_feature:nnnnn {Numbers} {ResetAll} {} {}
43 {
44   +tnum,-tnum,
45   +pnum,-pnum,
46   +onum,-onum,
47   +lnum,-lnum,
48   +zero,-zero,
49   +anum,-anum,
50 }
51 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Monospaced} {tnum} {tnum} {+pnum,-pnum}
52 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Proportional} {pnum} {pnum} {+tnum,-tnum}
53 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Lowercase} {onum} {onum} {+lnum,-lnum}
54 \@@_define_opentype_onoffreset:nnnnn {Numbers} {Uppercase} {lnum} {lnum} {+onum,-onum}
55 \@@_define_opentype_onoffreset:nnnnn {Numbers} {SlashedZero} {zero} {zero} {}
56 \aliasfontfeatureoption {Numbers} {Monospaced} {Tabular}
57 \aliasfontfeatureoption {Numbers} {Lowercase} {OldStyle}
58 \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.

```

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

```

2.4 Vertical position

```

60 \@@_define_opentype_feature_group:n {VerticalPosition}
61 \@@_define_opentype_feature:nnnnn {VerticalPosition} {ResetAll} {} {}
62 {
63   +sups,-sups,
64   +subs,-subs,
65   +ordn,-ordn,
66   +numr,-numr,
67   +dnom,-dnom,
68   +sinf,-sinf,
69 }
70 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Superior} {sups} {sups} {+}
71 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Inferior} {subs} {subs} {+}
72 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Ordinal} {ordn} {ordn} {+}
73 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Numerator} {numr} {numr} {+}
74 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {Denominator} {dnom} {dnom} {+}
75 \@@_define_opentype_onoffreset:nnnnn {VerticalPosition} {ScientificInferior} {sinf} {sinf} {+}

```

2.5 Contextuals


```

76 \@@_define_opentype_feature_group:n {Contextuals}
77 \@@_define_opentype_feature:nnnnn {Contextuals} {ResetAll} {} {}
78 {
79 +csw, -csw,
80 +calt, -calt,
81 +init, -init,
82 +fina, -fina,
83 +falt, -falt,
84 +medi, -medi,
85 }
86 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Swash} {csw} {csw} {}
87 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Alternate} {calt} {calt} {}
88 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordInitial} {init} {init} {}
89 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {WordFinal} {fina} {fina} {}
90 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {LineFinal} {falt} {falt} {}
91 \@@_define_opentype_onoffreset:nnnnn {Contextuals} {Inner} {medi} {medi} {}

```

2.6 Diacritics

```

92 \@@_define_opentype_feature_group:n {Diacritics}
93 \@@_define_opentype_feature:nnnnn {Diacritics} {ResetAll} {} {}
94 {
95 +mark, -mark,
96 +mkmk, -mkmk,
97 +abvm, -abvm,
98 +blwm, -blwm,
99 }
100 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToBase} {mark} {mark} {}
101 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {MarkToMark} {mkmk} {mkmk} {}
102 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {AboveBase} {abvm} {abvm} {}
103 \@@_define_opentype_onoffreset:nnnnn {Diacritics} {BelowBase} {blwm} {blwm} {}

```

2.7 Kerning

```

104 \@@_define_opentype_feature_group:n {Kerning}
105 \@@_define_opentype_feature:nnnnn {Kerning} {ResetAll} {} {}
106 {
107 +csp, -csp,
108 +kern, -kern,
109 }
110 \@@_define_opentype_onoffreset:nnnnn {Kerning} {Uppercase} {csp} {csp} {}
111 \@@_define_opentype_feature:nnnnn {Kerning} {On} {kern} {+kern} {-kern}
112 \@@_define_opentype_feature:nnnnn {Kerning} {Off} {kern} {-kern} {+kern}
113 \@@_define_opentype_feature:nnnnn {Kerning} {Reset} {} {} {+kern, -kern}

```

2.8 Fractions

```

114 \@@_define_opentype_feature_group:n {Fractions}
115 \@@_define_opentype_feature:nnnnn {Fractions} {ResetAll} {} {}
116 {
117 +frac, -frac,
118 +afrc, -afrc,
119 }

```

```

120 \@@_define_opentype_feature:nnnnn {Fractions} {On} {frac} {+frac} {}
121 \@@_define_opentype_feature:nnnnn {Fractions} {Off} {frac} {-frac} {}
122 \@@_define_opentype_feature:nnnnn {Fractions} {Reset} {} {} {+frac,-frac}
123 \@@_define_opentype_onoffreset:nnnnn {Fractions} {Alternate} {afrc} {afrc} {-frac}

124 \@@_define_opentype_feature_group:n {LocalForms}
125 \@@_define_opentype_feature:nnnnn {LocalForms} {On} {locl} {+locl} {}
126 \@@_define_opentype_feature:nnnnn {LocalForms} {Off} {locl} {-locl} {}
127 \@@_define_opentype_feature:nnnnn {LocalForms} {Reset} {} {} {+locl,-locl}

```

2.9 Style

```

128 \@@_define_opentype_feature_group:n {Style}
129 \@@_define_opentype_feature:nnnnn {Style} {ResetAll} {} {}
130 {
131   +salt,-salt,
132   +ital,-ital,
133   +ruby,-ruby,
134   +swsh,-swsh,
135   +hist,-hist,
136   +titl,-titl,
137   +hkna,-hkna,
138   +vkna,-vkna,
139   +ssty=0,-ssty=0,
140   +ssty=1,-ssty=1,
141 }

142 \@@_define_opentype_onoffreset:nnnnn {Style} {Alternate} {salt} {salt} {}
143 \@@_define_opentype_onoffreset:nnnnn {Style} {Italic} {ital} {ital} {}
144 \@@_define_opentype_onoffreset:nnnnn {Style} {Ruby} {ruby} {ruby} {}
145 \@@_define_opentype_onoffreset:nnnnn {Style} {Swash} {swsh} {swsh} {}
146 \@@_define_opentype_onoffreset:nnnnn {Style} {Cursive} {swsh} {curs} {}
147 \@@_define_opentype_onoffreset:nnnnn {Style} {Historic} {hist} {hist} {}
148 \@@_define_opentype_onoffreset:nnnnn {Style} {Titling} {titl} {titl} {}
149 \@@_define_opentype_onoffreset:nnnnn {Style} {TitlingCaps} {titl} {titl} {} % backwards
150 \@@_define_opentype_onoffreset:nnnnn {Style} {HorizontalKana} {hkna} {hkna} {+vkna,+pkna}
151 \@@_define_opentype_onoffreset:nnnnn {Style} {VerticalKana} {vkna} {vkna} {+hkna,+pkna}
152 \@@_define_opentype_onoffreset:nnnnn {Style} {ProportionalKana} {pkna} {pkna} {+vkna,+hkna}
153 \@@_define_opentype_feature:nnnnn {Style} {MathScript} {ssty} {+ssty=0} {+ssty=1}
154 \@@_define_opentype_feature:nnnnn {Style} {MathScriptScript} {ssty} {+ssty=1} {+ssty=0}
155 \@@_define_opentype_onoffreset:nnnnn {Style} {Uppercase} {case} {case} {}

```

2.10 CJK shape

```

156 \@@_define_opentype_feature_group:n {CJKShape}
157 \@@_define_opentype_feature:nnnnn {CJKShape} {ResetAll} {} {}
158 {
159   +trad,-trad,
160   +smp1,-smp1,
161   +jp78,-jp78,
162   +jp83,-jp83,
163   +jp90,-jp90,
164   +jp04,-jp04,
165   +expt,-expt,

```

```

166     +nlck,-nlck,
167   }
168   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Traditional} {trad} {trad} {+smpl,+jp78,+jp8
169   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Simplified} {smpl} {smpl} {+trad,+jp78,+jp8
170   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1978} {jp78} {jp78} {+trad,+smpl,+jp8
171   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1983} {jp83} {jp83} {+trad,+smpl,+jp7
172   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS1990} {jp90} {jp90} {+trad,+smpl,+jp7
173   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {JIS2004} {jp04} {jp04} {+trad,+smpl,+jp7
174   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {Expert} {expt} {expt} {+trad,+smpl,+jp7
175   \@@_define_opentype_onoffreset:nnnnn {CJKShape} {NLC} {nlck} {nlck} {+trad,+smpl,+jp7

```

2.11 Character width

```

176   \@@_define_opentype_feature_group:n {CharacterWidth}
177   \@@_define_opentype_feature:nnnnn {CharacterWidth} {ResetAll} {} {}
178   {
179     +pwid,-pwid,
180     +fwid,-fwid,
181     +hwid,-hwid,
182     +twid,-twid,
183     +qwid,-qwid,
184     +palt,-palt,
185     +halt,-halt,
186   }
187   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Proportional} {pwid} {pwid} {
188   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Full} {fwid} {fwid} {
189   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Half} {hwid} {hwid} {
190   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Third} {twid} {twid} {
191   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {Quarter} {qwid} {qwid} {
192   \@@_define_opentype_onoffreset:nnnnn {CharacterWidth} {AlternateProportional} {palt} {palt} {
193   \@@_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?).

```

194   \@@_define_opentype_feature_group:n {Vertical}
195   \@@_define_opentype_onoffreset:nnnnn {Vertical} {RotatedGlyphs} {vrt2} {vrt2} {+vrtr,
196   \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternatesForRotation} {vrtr} {vrtr} {+vrt2}
197   \@@_define_opentype_onoffreset:nnnnn {Vertical} {Alternates} {vert} {vert} {+vrt2}
198   \@@_define_opentype_onoffreset:nnnnn {Vertical} {KanaAlternates} {vkna} {vkna} {+hkna}
199   \@@_define_opentype_onoffreset:nnnnn {Vertical} {Kerning} {vkrn} {vkrn} {}
200   \@@_define_opentype_onoffreset:nnnnn {Vertical} {AlternateMetrics} {valt} {valt} {+vhal,
201   \@@_define_opentype_onoffreset:nnnnn {Vertical} {HalfMetrics} {vhal} {vhal} {+valt,
202   \@@_define_opentype_onoffreset:nnnnn {Vertical} {ProportionalMetrics} {vpal} {vpal} {+valt,

```

3 OpenType features that need numbering

3.1 Alternate

```

203   \@@_define_opentype_feature_group:n {Alternate}

```

```

204 \keys_define:nn {fontspec-opentype}
205 {
206   Alternate .default:n = {0} ,
207   Alternate .groups:n = {opentype},
208   Alternate / unknown .code:n =
209   {
210     \clist_map_inline:nn {#1}
211     { \@@_make_OT_feature:nnn {salt}{ +salt = ##1 }{} }
212   }
213 }
214 <*LU>
215 \keys_define:nn {fontspec-opentype}
216 {
217   Alternate / Random .code:n =
218   { \@@_make_OT_feature:nnn {salt}{ +salt = random }{} } ,
219 }
220 </LU>
221 \aliasfontfeature{Alternate}{StylisticAlternates}

```

3.2 Variant / StylisticSet

```

222 \@@_define_opentype_feature_group:n {Variant}
223 \keys_define:nn {fontspec-opentype}
224 {
225   Variant .default:n = {0} ,
226   Variant .groups:n = {opentype} ,
227   Variant / unknown .code:n =
228   {
229     \clist_map_inline:nn {#1}
230     {
231       \@@_make_OT_feature:xxx { ss \two@digits {##1} } { +ss \two@digits {##1} } {}
232     }
233   }
234 }
235 \aliasfontfeature{Variant}{StylisticSet}

```

3.3 CharacterVariant

```

236 \@@_define_opentype_feature_group:n {CharacterVariant}
237 \use:x
238 {
239   \cs_new:Npn \exp_not:N \fontspec_parse_cv:w
240   ##1 \c_colon_str ##2 \c_colon_str ##3 \exp_not:N \q_nil
241   {
242     \@@_make_OT_feature:xxx
243     { cv \exp_not:N \two@digits {##1} }
244     { +cv \exp_not:N \two@digits {##1} = ##2 } {}
245   }
246   \keys_define:nn {fontspec-opentype}
247   {
248     CharacterVariant / unknown .code:n =
249     {

```

```

250         \clist_map_inline:nn {##1}
251         {
252             \exp_not:N \fontspec_parse_cv:w
253             #####1 \c_colon_str 0 \c_colon_str \exp_not:N \q_nil
254         }
255     }
256 }
257 }

```

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

3.4 Annotation

```

258 \@@_define_opentype_feature_group:n {Annotation}
259 \keys_define:nn {fontspec-opentype}
260 {
261     Annotation .default:n = {0} ,
262     Annotation .groups:n = {opentype},
263     Annotation / unknown .code:n =
264     {
265         \@@_make_OT_feature:nnn {nalt} {+nalt=#1} {}
266     }
267 }

```

3.5 Ornament

```

268 \@@_define_opentype_feature_group:n {Ornament}
269 \keys_define:nn {fontspec-opentype}
270 {
271     Ornament .default:n = {0} ,
272     Ornament .groups:n = {opentype},
273     Ornament / unknown .code:n =
274     {
275         \@@_make_OT_feature:nnn {ornm} {+ornm=#1} {}
276     }
277 }

```

4 Script and Language

4.1 Script

```

278 \keys_define:nn {fontspec-opentype}
279 {
280     Script .choice: ,
281     Script .groups:n = {opentype} ,
282 }
283 \cs_new:Nn \fontspec_new_script:nn
284 {
285     \keys_define:nn {fontspec-opentype} { Script / #1 .code:n =
286     {
287 <debug>\typeout{Trying-[Script=#1]}
288         \bool_set_false:N \l_@@_scriptlang_exist_bool
289         \clist_map_inline:nn {#2}
290         {

```

```

291         \exp_args:No \@@_check_script:NnT \l_@@_fontface_cs_tl {####1}
292         {
293 <debug>\typeout{Script-tag-found:~####1}
294         \tl_set:Nn \l_@@_script_name_tl {#1}
295         \tl_set:Nn \l_@@_script_tl {####1}
296         \int_set:Nn \l_@@_script_int {\l_@@_strnum_int}
297         \bool_set_true:N \l_@@_scriptlang_exist_bool
298         \tl_gset:Nx \g_@@_single_feat_tl { script=####1 }
299         \clist_map_break:
300     }
301 }
302
303 \bool_if:NF \l_@@_scriptlang_exist_bool
304 {
305 <debug>\typeout{Script-not-found!}
306     \bool_if:nF { \str_if_eq_p:ee {#1} {CustomDefault} }
307     {
308         \tl_clear:N \l_@@_script_name_tl
309         \@@_warning:nxx {no-script} {\l_fontspeg_fontname_tl} {#1}
310     }
311
312     \bool_if:nF
313     {
314         \str_if_eq_p:ee {#1} {Default} ||
315         \str_if_eq_p:ee {#1} {Latin} ||
316         \str_if_eq_p:ee {#1} {CustomDefault}
317     }
318     {
319         \keys_set:nn {fontspec-opentype} { Script = CustomDefault }
320     }
321 }
322 }
323 }
324 }

```

4.2 Language

```

325 \keys_define:nn {fontspec-opentype}
326 {
327     Language .choice: ,
328     Language .groups:n = {opentype} ,
329 }
330 \cs_new:Nn \fontspec_new_lang:nn
331 {
332     \keys_define:nn {fontspec-opentype} { Language / #1 .code:n =
333     {
334         \bool_set_false:N \l_@@_scriptlang_exist_bool
335         \clist_map_inline:nn {#2}
336         {
337             \exp_args:No \@@_check_lang:NnTF \l_@@_fontface_cs_tl {####1}
338             {
339                 \tl_set:Nn \l_@@_lang_tl {####1}

```

```

340         \int_set:Nn \l_@@_language_int {\l_@@_strnum_int}
341         \tl_gset:Nx \g_@@_single_feat_tl { language=###1 }
342         \bool_set_true:N \l_@@_scriptlang_exist_bool
343         \clist_map_break:
344     }
345 }
346 \bool_if:NF \l_@@_scriptlang_exist_bool
347 {
348     \@@_warning:nx {language-not-exist} {#1}
349     \keys_set:nn {fontspec-opentype} { Language = Default }
350 }
351 }
352 }
353 }

```

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 X_YTeX-specific thing.

```

354 \@@_keys_define_code:nnn {fontspec-opentype} { Language / Default }
355 {
356     \tl_set:Nn \l_@@_lang_tl {dflt}
357     \int_zero:N \l_@@_language_int
358     \tl_gset:Nn \g_@@_single_feat_tl { language=dflt }
359 }

```

5 Backwards compatibility

```

360 \cs_new:Nn \@@_ot_compat:nn
361 {
362     \aliasfontfeatureoption {#1} {#20ff} {No#2}
363 }
364 \@@_ot_compat:nn {Ligatures} {Rare}
365 \@@_ot_compat:nn {Ligatures} {Required}
366 \@@_ot_compat:nn {Ligatures} {Common}
367 \@@_ot_compat:nn {Ligatures} {Discretionary}
368 \@@_ot_compat:nn {Ligatures} {Contextual}
369 \@@_ot_compat:nn {Ligatures} {Historic}
370 \@@_ot_compat:nn {Numbers} {SlashedZero}
371 \@@_ot_compat:nn {Contextuals} {Swash}
372 \@@_ot_compat:nn {Contextuals} {Alternate}
373 \@@_ot_compat:nn {Contextuals} {WordInitial}
374 \@@_ot_compat:nn {Contextuals} {WordFinal}
375 \@@_ot_compat:nn {Contextuals} {LineFinal}
376 \@@_ot_compat:nn {Contextuals} {Inner}
377 \@@_ot_compat:nn {Diacritics} {MarkToBase}
378 \@@_ot_compat:nn {Diacritics} {MarkToMark}
379 \@@_ot_compat:nn {Diacritics} {AboveBase}
380 \@@_ot_compat:nn {Diacritics} {BelowBase}

```

File XV

fontspec-code-scripts.dtx

1 Font script definitions

```
1 \newfontscript{Adlam}{adlm}
2 \newfontscript{Ahom}{ahom}
3 \newfontscript{Anatolian~Hieroglyphs}{hluw}
4 \newfontscript{Arabic}{arab}
5 \newfontscript{Armenian}{armn}
6 \newfontscript{Avestan}{avst}
7 \newfontscript{Balinese}{bali}
8 \newfontscript{Bamum}{bamu}
9 \newfontscript{Bassa~Vah}{bass}
10 \newfontscript{Batak}{batk}
11 \newfontscript{Bengali}{bng2,beng}
12 \newfontscript{Bhaiksuki}{bhks}
13 \newfontscript{Bopomofo}{bopo}
14 \newfontscript{Brahmi}{brah}
15 \newfontscript{Braille}{brai}
16 \newfontscript{Buginese}{bugi}
17 \newfontscript{Buhid}{buhd}
18 \newfontscript{Byzantine~Music}{byzm}
19 \newfontscript{Canadian~Syllabics}{cans}
20 \newfontscript{Carian}{cari}
21 \newfontscript{Caucasian~Albanian}{aghb}
22 \newfontscript{Chakma}{cakm}
23 \newfontscript{Cham}{cham}
24 \newfontscript{Cherokee}{cher}
25 \newfontscript{CJK~Ideographic}{hani}
26 \newfontscript{Coptic}{copt}
27 \newfontscript{Cypriot~Syllabary}{cpri}
28 \newfontscript{Cyrillic}{cyrl}
29 \newfontscript{Default}{DFLT}
30 \newfontscript{CustomDefault}{latn,DFLT}
31 \newfontscript{Deseret}{dsrt}
32 \newfontscript{Devanagari}{dev2,deva}
33 \newfontscript{Dogra}{dogr}
34 \newfontscript{Duployan}{dupl}
35 \newfontscript{Egyptian~Hieroglyphs}{egyp}
36 \newfontscript{Elbasan}{elba}
37 \newfontscript{Ethiopic}{ethi}
38 \newfontscript{Georgian}{geor}
39 \newfontscript{Glagolitic}{glag}
40 \newfontscript{Gothic}{goth}
41 \newfontscript{Grantha}{gran}
42 \newfontscript{Greek}{grek}
43 \newfontscript{Gujarati}{gjr2,gujr}
44 \newfontscript{Gunjala~Gondi}{gong}
```


45 \newfontscript{Gurmukhi}{gur2,guru}
46 \newfontscript{Hangul~Jamo}{jamo}
47 \newfontscript{Hangul}{hang}
48 \newfontscript{Hanifi~Rohingya}{rohng}
49 \newfontscript{Hanunoo}{hano}
50 \newfontscript{Hatran}{hatr}
51 \newfontscript{Hebrew}{hebr}
52 \newfontscript{Hiragana~and~Katakana}{kana}
53 \newfontscript{Imperial~Aramaic}{armi}
54 \newfontscript{Inscriptional~Pahlavi}{phli}
55 \newfontscript{Inscriptional~Parthian}{prti}
56 \newfontscript{Javanese}{java}
57 \newfontscript{Kaithi}{kthi}
58 \newfontscript{Kannada}{knd2,knda}
59 \newfontscript{Kayah~Li}{kali}
60 \newfontscript{Kharosthi}{khar}
61 \newfontscript{Khmer}{khmr}
62 \newfontscript{Khojki}{khoj}
63 \newfontscript{Khudawadi}{sind}
64 \newfontscript{Lao}{lao~}
65 \newfontscript{Latin}{latn}
66 \newfontscript{Lepcha}{lepc}
67 \newfontscript{Limbu}{limb}
68 \newfontscript{Linear~A}{lina}
69 \newfontscript{Linear~B}{linb}
70 \newfontscript{Lisu}{lisu}
71 \newfontscript{Lycian}{lyci}
72 \newfontscript{Lydian}{lydi}
73 \newfontscript{Mahajani}{mahj}
74 \newfontscript{Makasar}{maka}
75 \newfontscript{Malayalam}{mlm2,mlym}
76 \newfontscript{Mandaic}{mand}
77 \newfontscript{Manichaeen}{mani}
78 \newfontscript{Marchen}{marc}
79 \newfontscript{Masaram Gondi}{gonm}
80 \newfontscript{Math}{math}
81 \newfontscript{Medefaidrin}{medf}
82 \newfontscript{Meitei~Mayek}{mtei}
83 \newfontscript{Mende~Kikakui}{mend}
84 \newfontscript{Meroitic~Cursive}{merc}
85 \newfontscript{Meroitic~Hieroglyphs}{mero}
86 \newfontscript{Miao}{plrd}
87 \newfontscript{Modi}{modi}
88 \newfontscript{Mongolian}{mong}
89 \newfontscript{Mro}{mroo}
90 \newfontscript{Multani}{mult}
91 \newfontscript{Musical~Symbols}{musc}
92 \newfontscript{Myanmar}{mym2,mymr}
93 \newfontscript{N'Ko}{nko~}
94 \newfontscript{Nabataean}{nbat}
95 \newfontscript{Newa}{newa}

96 \newfontscript{Nushu}{nshu}
 97 \newfontscript{Odia}{ory2,orya}
 98 \newfontscript{Ogham}{ogam}
 99 \newfontscript{Ol~Chiki}{olck}
 100 \newfontscript{Old~Italic}{ital}
 101 \newfontscript{Old~Hungarian}{hung}
 102 \newfontscript{Old~North~Arabian}{narb}
 103 \newfontscript{Old~Permic}{perm}
 104 \newfontscript{Old~Persian~Cuneiform}{xpeo}
 105 \newfontscript{Old~Sogdian}{sogo}
 106 \newfontscript{Old~South~Arabian}{sarb}
 107 \newfontscript{Old~Turkic}{orkh}
 108 \newfontscript{Osage}{osge}
 109 \newfontscript{Osmanya}{osma}
 110 \newfontscript{Pahawh~Hmong}{hmng}
 111 \newfontscript{Palmyrene}{palm}
 112 \newfontscript{Pau~Cin~Hau}{pauc}
 113 \newfontscript{Phags~pa}{phag}
 114 \newfontscript{Phoenician}{phnx}
 115 \newfontscript{Psalter~Pahlavi}{phlp}
 116 \newfontscript{Rejang}{rjng}
 117 \newfontscript{Runic}{runr}
 118 \newfontscript{Samaritan}{samr}
 119 \newfontscript{Saurashtra}{saur}
 120 \newfontscript{Sharada}{shrd}
 121 \newfontscript{Shavian}{shaw}
 122 \newfontscript{Siddham}{sidd}
 123 \newfontscript{Sign~Writing}{sgnw}
 124 \newfontscript{Sinhala}{sinh}
 125 \newfontscript{Sogdian}{sogd}
 126 \newfontscript{Sora~Sompeng}{sora}
 127 \newfontscript{Sumero~Akkadian~Cuneiform}{xsux}
 128 \newfontscript{Sundanese}{sund}
 129 \newfontscript{Syloti~Nagri}{sylo}
 130 \newfontscript{Syriac}{syrc}
 131 \newfontscript{Tagalog}{tglg}
 132 \newfontscript{Tagbanwa}{tagb}
 133 \newfontscript{Tai~Le}{tale}
 134 \newfontscript{Tai~Lu}{talu}
 135 \newfontscript{Tai~Tham}{lana}
 136 \newfontscript{Tai~Viet}{tavl}
 137 \newfontscript{Takri}{takr}
 138 \newfontscript{Tamil}{tml2,taml}
 139 \newfontscript{Tangut}{tang}
 140 \newfontscript{Telugu}{tel2,telu}
 141 \newfontscript{Thaana}{thaa}
 142 \newfontscript{Thai}{thai}
 143 \newfontscript{Tibetan}{tibt}
 144 \newfontscript{Tifinagh}{tfng}
 145 \newfontscript{Tirhuta}{tirh}
 146 \newfontscript{Ugaritic~Cuneiform}{ugar}

```
147 \newfontscript{Vai}{vai~}  
148 \newfontscript{Warang~Citi}{wara}  
149 \newfontscript{Yi}{yi~~}  
150 \newfontscript{Zanabazar~Square}{zanb}
```

For convenience or backwards compatibility:

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

File XVI

fontspec-code-lang.dtx

1 Font language definitions

```
1 \newfontlanguage{Abaza}{ABA}
2 \newfontlanguage{Abkhazian}{ABK}
3 \newfontlanguage{Adyghe}{ADY}
4 \newfontlanguage{Afrikaans}{AFK}
5 \newfontlanguage{Afar}{AFR}
6 \newfontlanguage{Agaw}{AGW}
7 \newfontlanguage{Altai}{ALT}
8 \newfontlanguage{Amharic}{AMH}
9 \newfontlanguage{Arabic}{ARA}
10 \newfontlanguage{Aari}{ARI}
11 \newfontlanguage{Arakanese}{ARK}
12 \newfontlanguage{Assamese}{ASM}
13 \newfontlanguage{Athapaskan}{ATH}
14 \newfontlanguage{Avar}{AVR}
15 \newfontlanguage{Awadhi}{AWA}
16 \newfontlanguage{Aymara}{AYM}
17 \newfontlanguage{Azeri}{AZE}
18 \newfontlanguage{Badaga}{BAD}
19 \newfontlanguage{Baghelkhandi}{BAG}
20 \newfontlanguage{Balkar}{BAL}
21 \newfontlanguage{Baule}{BAU}
22 \newfontlanguage{Berber}{BBR}
23 \newfontlanguage{Bench}{BCH}
24 \newfontlanguage{Bible~Cree}{BCR}
25 \newfontlanguage{Belarussian}{BEL}
26 \newfontlanguage{Bemba}{BEM}
27 \newfontlanguage{Bengali}{BEN}
28 \newfontlanguage{Bulgarian}{BGR}
29 \newfontlanguage{Bhili}{BHI}
30 \newfontlanguage{Bhojpurī}{BHO}
31 \newfontlanguage{Bikol}{BIK}
32 \newfontlanguage{Bilen}{BIL}
33 \newfontlanguage{Blackfoot}{BKF}
34 \newfontlanguage{Balochi}{BLI}
35 \newfontlanguage{Balante}{BLN}
36 \newfontlanguage{Balti}{BLT}
37 \newfontlanguage{Bambara}{BMB}
38 \newfontlanguage{Bamileke}{BML}
39 \newfontlanguage{Breton}{BRE}
40 \newfontlanguage{Brahui}{BRH}
41 \newfontlanguage{Braj~Bhasha}{BRI}
42 \newfontlanguage{Burmese}{BRM}
43 \newfontlanguage{Bashkir}{BSH}
44 \newfontlanguage{Beti}{BTI}
```

45 \newfontlanguage{Catalan}{CAT}
46 \newfontlanguage{Cebuano}{CEB}
47 \newfontlanguage{Chechen}{CHE}
48 \newfontlanguage{Chaha~Gurage}{CHG}
49 \newfontlanguage{Chattisgarhi}{CHH}
50 \newfontlanguage{Chichewa}{CHI}
51 \newfontlanguage{Chukchi}{CHK}
52 \newfontlanguage{Chipewyan}{CHP}
53 \newfontlanguage{Cherokee}{CHR}
54 \newfontlanguage{Chuvash}{CHU}
55 \newfontlanguage{Comorian}{CMR}
56 \newfontlanguage{Coptic}{COP}
57 \newfontlanguage{Cree}{CRE}
58 \newfontlanguage{Carrier}{CRR}
59 \newfontlanguage{Crimean~Tatar}{CRT}
60 \newfontlanguage{Church~Slavonic}{CSL}
61 \newfontlanguage{Czech}{CSY}
62 \newfontlanguage{Danish}{DAN}
63 \newfontlanguage{Dargwa}{DAR}
64 \newfontlanguage{Woods~Cree}{DCR}
65 \newfontlanguage{German}{DEU}
66 \newfontlanguage{Dogri}{DGR}
67 \newfontlanguage{Divehi}{DIV}
68 \newfontlanguage{Djerma}{DJR}
69 \newfontlanguage{Dangme}{DNG}
70 \newfontlanguage{Dinka}{DNK}
71 \newfontlanguage{Dungan}{DUN}
72 \newfontlanguage{Dzongkha}{DZN}
73 \newfontlanguage{Ebira}{EBI}
74 \newfontlanguage{Eastern~Cree}{ECR}
75 \newfontlanguage{Edo}{EDO}
76 \newfontlanguage{Efik}{EFI}
77 \newfontlanguage{Greek}{ELL}
78 \newfontlanguage{English}{ENG}
79 \newfontlanguage{Erzya}{ERZ}
80 \newfontlanguage{Spanish}{ESP}
81 \newfontlanguage{Estonian}{ETI}
82 \newfontlanguage{Basque}{EUQ}
83 \newfontlanguage{Evenki}{EVK}
84 \newfontlanguage{Even}{EVN}
85 \newfontlanguage{Ewe}{EWE}
86 \newfontlanguage{French~Antillean}{FAN}
87 \newfontlanguage{Farsi}{FAR}
88 \newfontlanguage{Parsi}{FAR}
89 \newfontlanguage{Persian}{FAR}
90 \newfontlanguage{Finnish}{FIN}
91 \newfontlanguage{Fijian}{FJI}
92 \newfontlanguage{Flemish}{FLE}
93 \newfontlanguage{Forest~Nenets}{FNE}
94 \newfontlanguage{Fon}{FON}
95 \newfontlanguage{Faroese}{FOS}

96 \newfontlanguage{French}{FRA}
 97 \newfontlanguage{Frisian}{FRI}
 98 \newfontlanguage{Friulian}{FRL}
 99 \newfontlanguage{Futa}{FTA}
 100 \newfontlanguage{Fulani}{FUL}
 101 \newfontlanguage{Ga}{GAD}
 102 \newfontlanguage{Gaelic}{GAE}
 103 \newfontlanguage{Gagauz}{GAG}
 104 \newfontlanguage{Galician}{GAL}
 105 \newfontlanguage{Garshuni}{GAR}
 106 \newfontlanguage{Garhwali}{GAW}
 107 \newfontlanguage{Ge'ez}{GEZ}
 108 \newfontlanguage{Gilyak}{GIL}
 109 \newfontlanguage{Gumuz}{GMZ}
 110 \newfontlanguage{Gondi}{GON}
 111 \newfontlanguage{Greenlandic}{GRN}
 112 \newfontlanguage{Garo}{GRO}
 113 \newfontlanguage{Guarani}{GUA}
 114 \newfontlanguage{Gujarati}{GUJ}
 115 \newfontlanguage{Haitian}{HAI}
 116 \newfontlanguage{Halam}{HAL}
 117 \newfontlanguage{Harauti}{HAR}
 118 \newfontlanguage{Hausa}{HAU}
 119 \newfontlanguage{Hawaiin}{HAW}
 120 \newfontlanguage{Hammer-Banna}{HBN}
 121 \newfontlanguage{Hiligaynon}{HIL}
 122 \newfontlanguage{Hindi}{HIN}
 123 \newfontlanguage{High-Mari}{HMA}
 124 \newfontlanguage{Hindko}{HND}
 125 \newfontlanguage{Ho}{HO}
 126 \newfontlanguage{Harari}{HRI}
 127 \newfontlanguage{Croatian}{HRV}
 128 \newfontlanguage{Hungarian}{HUN}
 129 \newfontlanguage{Armenian}{HYE}
 130 \newfontlanguage{Igbo}{IBO}
 131 \newfontlanguage{Ijo}{IJO}
 132 \newfontlanguage{Ilokano}{ILO}
 133 \newfontlanguage{Indonesian}{IND}
 134 \newfontlanguage{Ingush}{ING}
 135 \newfontlanguage{Inuktitut}{INU}
 136 \newfontlanguage{Irish}{IRI}
 137 \newfontlanguage{Irish-Traditional}{IRT}
 138 \newfontlanguage{Icelandic}{ISL}
 139 \newfontlanguage{Inari-Sami}{ISM}
 140 \newfontlanguage{Italian}{ITA}
 141 \newfontlanguage{Hebrew}{IWR}
 142 \newfontlanguage{Javanese}{JAV}
 143 \newfontlanguage{Yiddish}{JII}
 144 \newfontlanguage{Japanese}{JAN}
 145 \newfontlanguage{Judezmo}{JUD}
 146 \newfontlanguage{Jula}{JUL}

147 \newfontlanguage{Kabardian}{KAB}
148 \newfontlanguage{Kachchi}{KAC}
149 \newfontlanguage{Kalenjin}{KAL}
150 \newfontlanguage{Kannada}{KAN}
151 \newfontlanguage{Karachay}{KAR}
152 \newfontlanguage{Georgian}{KAT}
153 \newfontlanguage{Kazakh}{KAZ}
154 \newfontlanguage{Kebena}{KEB}
155 \newfontlanguage{Khutsuri~Georgian}{KGE}
156 \newfontlanguage{Khakass}{KHA}
157 \newfontlanguage{Khanty-Kazim}{KHK}
158 \newfontlanguage{Khmer}{KHM}
159 \newfontlanguage{Khanty-Shurishkar}{KHS}
160 \newfontlanguage{Khanty-Vakhi}{KHV}
161 \newfontlanguage{Khowar}{KHW}
162 \newfontlanguage{Kikuyu}{KIK}
163 \newfontlanguage{Kirghiz}{KIR}
164 \newfontlanguage{Kisii}{KIS}
165 \newfontlanguage{Kokni}{KKN}
166 \newfontlanguage{Kalmyk}{KLM}
167 \newfontlanguage{Kamba}{KMB}
168 \newfontlanguage{Kumaoni}{KMN}
169 \newfontlanguage{Komo}{KMO}
170 \newfontlanguage{Komso}{KMS}
171 \newfontlanguage{Kanuri}{KNR}
172 \newfontlanguage{Kodagu}{KOD}
173 \newfontlanguage{Korean-Old-Hangul}{KOH}
174 \newfontlanguage{Konkani}{KOK}
175 \newfontlanguage{Kikongo}{KON}
176 \newfontlanguage{Komi-Permyak}{KOP}
177 \newfontlanguage{Korean}{KOR}
178 \newfontlanguage{Komi-Zyrian}{KOZ}
179 \newfontlanguage{Kpelle}{KPL}
180 \newfontlanguage{Krio}{KRI}
181 \newfontlanguage{Karakalpak}{KRK}
182 \newfontlanguage{Karelian}{KRL}
183 \newfontlanguage{Karaim}{KRM}
184 \newfontlanguage{Karen}{KRN}
185 \newfontlanguage{Koorete}{KRT}
186 \newfontlanguage{Kashmiri}{KSH}
187 \newfontlanguage{Khasi}{KSI}
188 \newfontlanguage{Kildin-Sami}{KSM}
189 \newfontlanguage{Kui}{KUI}
190 \newfontlanguage{Kulvi}{KUL}
191 \newfontlanguage{Kumyk}{KUM}
192 \newfontlanguage{Kurdish}{KUR}
193 \newfontlanguage{Kurukh}{KUU}
194 \newfontlanguage{Kuy}{KUY}
195 \newfontlanguage{Koryak}{KYK}
196 \newfontlanguage{Ladin}{LAD}
197 \newfontlanguage{Lahuli}{LAH}

198 \newfontlanguage{Lak}{LAK}
199 \newfontlanguage{Lambani}{LAM}
200 \newfontlanguage{Lao}{LAO}
201 \newfontlanguage{Latin}{LAT}
202 \newfontlanguage{Laz}{LAZ}
203 \newfontlanguage{L-Cree}{LCR}
204 \newfontlanguage{Ladakhi}{LDK}
205 \newfontlanguage{Lezgi}{LEZ}
206 \newfontlanguage{Lingala}{LIN}
207 \newfontlanguage{Low~Mari}{LMA}
208 \newfontlanguage{Limbu}{LMB}
209 \newfontlanguage{Lomwe}{LMW}
210 \newfontlanguage{Lower~Sorbian}{LSB}
211 \newfontlanguage{Lule~Sami}{LSM}
212 \newfontlanguage{Lithuanian}{LTH}
213 \newfontlanguage{Luba}{LUB}
214 \newfontlanguage{Luganda}{LUG}
215 \newfontlanguage{Luhya}{LUH}
216 \newfontlanguage{Luo}{LUO}
217 \newfontlanguage{Latvian}{LVI}
218 \newfontlanguage{Majang}{MAJ}
219 \newfontlanguage{Makua}{MAK}
220 \newfontlanguage{Malayalam~Traditional}{MAL}
221 \newfontlanguage{Mansi}{MAN}
222 \newfontlanguage{Marathi}{MAR}
223 \newfontlanguage{Marwari}{MAW}
224 \newfontlanguage{Mbundu}{MBN}
225 \newfontlanguage{Manchu}{MCH}
226 \newfontlanguage{Moose~Cree}{MCR}
227 \newfontlanguage{Mende}{MDE}
228 \newfontlanguage{Me'en}{MEN}
229 \newfontlanguage{Mizo}{MIZ}
230 \newfontlanguage{Macedonian}{MKD}
231 \newfontlanguage{Male}{MLE}
232 \newfontlanguage{Malagasy}{MLG}
233 \newfontlanguage{Malinke}{MLN}
234 \newfontlanguage{Malayalam~Reformed}{MLR}
235 \newfontlanguage{Malay}{MLY}
236 \newfontlanguage{Mandinka}{MND}
237 \newfontlanguage{Mongolian}{MNG}
238 \newfontlanguage{Manipuri}{MNI}
239 \newfontlanguage{Maninka}{MNK}
240 \newfontlanguage{Manx~Gaelic}{MNX}
241 \newfontlanguage{Moksha}{MOK}
242 \newfontlanguage{Moldavian}{MOL}
243 \newfontlanguage{Mon}{MON}
244 \newfontlanguage{Moroccan}{MOR}
245 \newfontlanguage{Maori}{MRI}
246 \newfontlanguage{Maithili}{MTH}
247 \newfontlanguage{Maltese}{MTS}
248 \newfontlanguage{Mundari}{MUN}

249 \newfontlanguage{Naga-Assamese}{NAG}
250 \newfontlanguage{Nanai}{NAN}
251 \newfontlanguage{Naskapi}{NAS}
252 \newfontlanguage{N-Cree}{NCR}
253 \newfontlanguage{Ndebele}{NDB}
254 \newfontlanguage{Ndonga}{NDG}
255 \newfontlanguage{Nepali}{NEP}
256 \newfontlanguage{Newari}{NEW}
257 \newfontlanguage{Nagari}{NGR}
258 \newfontlanguage{Norway-House-Cree}{NHC}
259 \newfontlanguage{Nisi}{NIS}
260 \newfontlanguage{Niuean}{NIU}
261 \newfontlanguage{Nkole}{NKL}
262 \newfontlanguage{N'ko}{NKO}
263 \newfontlanguage{Dutch}{NLD}
264 \newfontlanguage{Nogai}{NOG}
265 \newfontlanguage{Norwegian}{NOR}
266 \newfontlanguage{Northern-Sami}{NSM}
267 \newfontlanguage{Northern-Tai}{NTA}
268 \newfontlanguage{Esperanto}{NTO}
269 \newfontlanguage{Nynorsk}{NYN}
270 \newfontlanguage{Oji-Cree}{OCR}
271 \newfontlanguage{Ojibway}{OBJ}
272 \newfontlanguage{Oriya}{ORI}
273 \newfontlanguage{Oromo}{ORO}
274 \newfontlanguage{Ossetian}{OSS}
275 \newfontlanguage{Palestinian-Aramaic}{PAA}
276 \newfontlanguage{Pali}{PAL}
277 \newfontlanguage{Punjabi}{PAN}
278 \newfontlanguage{Palpa}{PAP}
279 \newfontlanguage{Pashto}{PAS}
280 \newfontlanguage{Polytonic-Greek}{PGR}
281 \newfontlanguage{Pilipino}{PIL}
282 \newfontlanguage{Palaung}{PLG}
283 \newfontlanguage{Polish}{PLK}
284 \newfontlanguage{Provençal}{PRO}
285 \newfontlanguage{Portuguese}{PTG}
286 \newfontlanguage{Chin}{QIN}
287 \newfontlanguage{Rajasthani}{RAJ}
288 \newfontlanguage{R-Cree}{RCR}
289 \newfontlanguage{Russian-Buriat}{RBU}
290 \newfontlanguage{Riang}{RIA}
291 \newfontlanguage{Rhaeto-Romanic}{RMS}
292 \newfontlanguage{Romanian}{ROM}
293 \newfontlanguage{Romany}{ROY}
294 \newfontlanguage{Rusyn}{RSY}
295 \newfontlanguage{Ruanda}{RUA}
296 \newfontlanguage{Russian}{RUS}
297 \newfontlanguage{Sadri}{SAD}
298 \newfontlanguage{Sanskrit}{SAN}
299 \newfontlanguage{Santali}{SAT}

300 \newfontlanguage{Sayisi}{SAY}
301 \newfontlanguage{Sekota}{SEK}
302 \newfontlanguage{Selkup}{SEL}
303 \newfontlanguage{Sango}{SGO}
304 \newfontlanguage{Shan}{SHN}
305 \newfontlanguage{Sibe}{SIB}
306 \newfontlanguage{Sidamo}{SID}
307 \newfontlanguage{Silte~Gurage}{SIG}
308 \newfontlanguage{Skolt~Sami}{SKS}
309 \newfontlanguage{Slovak}{SKY}
310 \newfontlanguage{Slavey}{SLA}
311 \newfontlanguage{Slovenian}{SLV}
312 \newfontlanguage{Somali}{SML}
313 \newfontlanguage{Samoan}{SMO}
314 \newfontlanguage{Sena}{SNA}
315 \newfontlanguage{Sindhi}{SND}
316 \newfontlanguage{Sinhalese}{SNH}
317 \newfontlanguage{Soninke}{SNK}
318 \newfontlanguage{Sodo~Gurage}{SOG}
319 \newfontlanguage{Sotho}{SOT}
320 \newfontlanguage{Albanian}{SQI}
321 \newfontlanguage{Serbian}{SRB}
322 \newfontlanguage{Saraiki}{SRK}
323 \newfontlanguage{Serer}{SRR}
324 \newfontlanguage{South~Slavey}{SSL}
325 \newfontlanguage{Southern~Sami}{SSM}
326 \newfontlanguage{Suri}{SUR}
327 \newfontlanguage{Svan}{SVA}
328 \newfontlanguage{Swedish}{SVE}
329 \newfontlanguage{Swadaya~Aramaic}{SWA}
330 \newfontlanguage{Swahili}{SWK}
331 \newfontlanguage{Swazi}{SWZ}
332 \newfontlanguage{Sutu}{SXT}
333 \newfontlanguage{Syriac}{SYR}
334 \newfontlanguage{Tabasaran}{TAB}
335 \newfontlanguage{Tajiki}{TAJ}
336 \newfontlanguage{Tamil}{TAM}
337 \newfontlanguage{Tatar}{TAT}
338 \newfontlanguage{TH~Cree}{TCR}
339 \newfontlanguage{Telugu}{TEL}
340 \newfontlanguage{Tongan}{TGN}
341 \newfontlanguage{Tigre}{TGR}
342 \newfontlanguage{Tigrinya}{TGY}
343 \newfontlanguage{Thai}{THA}
344 \newfontlanguage{Tahitian}{THT}
345 \newfontlanguage{Tibetan}{TIB}
346 \newfontlanguage{Turkish}{TRK, TUR}
347 \newfontlanguage{Turkmen}{TKM}
348 \newfontlanguage{Temne}{TMN}
349 \newfontlanguage{Tswana}{TNA}
350 \newfontlanguage{Tundra~Nenets}{TNE}

```

351 \newfontlanguage{Tonga}{TNG}
352 \newfontlanguage{Todo}{TOD}
353 \newfontlanguage{Tsonga}{TSG}
354 \newfontlanguage{Turoyo-Aramaic}{TUA}
355 \newfontlanguage{Tulu}{TUL}
356 \newfontlanguage{Tuvin}{TUV}
357 \newfontlanguage{Twí}{TWI}
358 \newfontlanguage{Udmurt}{UDM}
359 \newfontlanguage{Ukrainian}{UKR}
360 \newfontlanguage{Urdu}{URD}
361 \newfontlanguage{Upper-Sorbian}{USB}
362 \newfontlanguage{Uyghur}{UYG}
363 \newfontlanguage{Uzbek}{UZB}
364 \newfontlanguage{Venda}{VEN}
365 \newfontlanguage{Vietnamese}{VIT}
366 \newfontlanguage{Wa}{WA}
367 \newfontlanguage{Wagdi}{WAG}
368 \newfontlanguage{West-Cree}{WCR}
369 \newfontlanguage{Welsh}{WEL}
370 \newfontlanguage{Wolof}{WLF}
371 \newfontlanguage{Tai-Lue}{XBD}
372 \newfontlanguage{Xhosa}{XHS}
373 \newfontlanguage{Yakut}{YAK}
374 \newfontlanguage{Yoruba}{YBA}
375 \newfontlanguage{Y-Cree}{YCR}
376 \newfontlanguage{Yi-Classic}{YIC}
377 \newfontlanguage{Yi-Modern}{YIM}
378 \newfontlanguage{Chinese-Hong-Kong}{ZHH}
379 \newfontlanguage{Chinese-Phonetic}{ZHP}
380 \newfontlanguage{Chinese-Simplified}{ZHS}
381 \newfontlanguage{Chinese-Traditional}{ZHT}
382 \newfontlanguage{Zande}{ZND}
383 \newfontlanguage{Zulu}{ZUL}

```

File XVII

fontspec-code-feat-aat.dtx

1 AAT feature definitions

These are only defined for X₃TeX.

1.1 Ligatures

```
1 \@@_define_aat_feature_group:n {Ligatures}
2 \@@_define_aat_feature:nnnn {Ligatures} {Required} {1} {0}
3 \@@_define_aat_feature:nnnn {Ligatures} {NoRequired} {1} {1}
4 \@@_define_aat_feature:nnnn {Ligatures} {Common} {1} {2}
5 \@@_define_aat_feature:nnnn {Ligatures} {NoCommon} {1} {3}
6 \@@_define_aat_feature:nnnn {Ligatures} {Rare} {1} {4}
7 \@@_define_aat_feature:nnnn {Ligatures} {NoRare} {1} {5}
8 \@@_define_aat_feature:nnnn {Ligatures} {Discretionary} {1} {4}
9 \@@_define_aat_feature:nnnn {Ligatures} {NoDiscretionary} {1} {5}
10 \@@_define_aat_feature:nnnn {Ligatures} {Logos} {1} {6}
11 \@@_define_aat_feature:nnnn {Ligatures} {NoLogos} {1} {7}
12 \@@_define_aat_feature:nnnn {Ligatures} {Rebus} {1} {8}
13 \@@_define_aat_feature:nnnn {Ligatures} {NoRebus} {1} {9}
14 \@@_define_aat_feature:nnnn {Ligatures} {Diphthong} {1} {10}
15 \@@_define_aat_feature:nnnn {Ligatures} {NoDiphthong} {1} {11}
16 \@@_define_aat_feature:nnnn {Ligatures} {Squared} {1} {12}
17 \@@_define_aat_feature:nnnn {Ligatures} {NoSquared} {1} {13}
18 \@@_define_aat_feature:nnnn {Ligatures} {AbbrevSquared} {1} {14}
19 \@@_define_aat_feature:nnnn {Ligatures} {NoAbbrevSquared} {1} {15}
20 \@@_define_aat_feature:nnnn {Ligatures} {Icelandic} {1} {32}
21 \@@_define_aat_feature:nnnn {Ligatures} {NoIcelandic} {1} {33}
```

Emulate CM extra ligatures.

```
22 \keys_define:nn {fontspec-aat}
23 {
24   Ligatures / TeX .code:n =
25   {
26     \tl_set:Nn \l_@@_mapping_tl { tex-text }
27   }
28 }
```

1.2 Letters

```
29 \@@_define_aat_feature_group:n {Letters}
30 \@@_define_aat_feature:nnnn {Letters} {Normal} {3} {0}
31 \@@_define_aat_feature:nnnn {Letters} {Uppercase} {3} {1}
32 \@@_define_aat_feature:nnnn {Letters} {Lowercase} {3} {2}
33 \@@_define_aat_feature:nnnn {Letters} {SmallCaps} {3} {3}
34 \@@_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.

```
35 \@@_define_aat_feature_group:n {Numbers}
36 \@@_define_aat_feature:nnnn {Numbers} {Monospaced} {6} {0}
37 \@@_define_aat_feature:nnnn {Numbers} {Proportional} {6} {1}
38 \@@_define_aat_feature:nnnn {Numbers} {Lowercase} {21} {0}
39 \@@_define_aat_feature:nnnn {Numbers} {OldStyle} {21} {0}
40 \@@_define_aat_feature:nnnn {Numbers} {Uppercase} {21} {1}
41 \@@_define_aat_feature:nnnn {Numbers} {Lining} {21} {1}
42 \@@_define_aat_feature:nnnn {Numbers} {SlashedZero} {14} {5}
43 \@@_define_aat_feature:nnnn {Numbers} {NoSlashedZero} {14} {4}
```

1.4 Contextuals

```
44 \@@_define_aat_feature_group:n {Contextuals}
45 \@@_define_aat_feature:nnnn {Contextuals} {WordInitial} {8} {0}
46 \@@_define_aat_feature:nnnn {Contextuals} {NoWordInitial} {8} {1}
47 \@@_define_aat_feature:nnnn {Contextuals} {WordFinal} {8} {2}
48 \@@_define_aat_feature:nnnn {Contextuals} {NoWordFinal} {8} {3}
49 \@@_define_aat_feature:nnnn {Contextuals} {LineInitial} {8} {4}
50 \@@_define_aat_feature:nnnn {Contextuals} {NoLineInitial} {8} {5}
51 \@@_define_aat_feature:nnnn {Contextuals} {LineFinal} {8} {6}
52 \@@_define_aat_feature:nnnn {Contextuals} {NoLineFinal} {8} {7}
53 \@@_define_aat_feature:nnnn {Contextuals} {Inner} {8} {8}
54 \@@_define_aat_feature:nnnn {Contextuals} {NoInner} {8} {9}
```

1.5 Diacritics

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

1.6 Vertical position

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

1.7 Fractions

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

1.8 Alternate

```
68 \@@_define_aat_feature_group:n { Alternate }
```

```

69 \keys_define:nn {fontspec-aat}
70 {
71   Alternate .default:n = {0} ,
72   Alternate / unknown .code:n =
73     {
74       \clist_map_inline:nn {#1}
75       {
76         \@@_make_AAT_feature:nn {17}{##1}
77       }
78     }
79 }

```

1.9 Variant / StylisticSet

```

80 \@@_define_aat_feature_group:n {Variant}
81 \keys_define:nn {fontspec-aat}
82 {
83   Variant .default:n = {0} ,
84   Variant / unknown .code:n =
85     {
86       \clist_map_inline:nn {#1}
87       { \@@_make_AAT_feature:nn {18}{##1} }
88     }
89 }
90 \aliasfontfeature{Variant}{StylisticSet}
91 \@@_define_aat_feature_group:n {Vertical}
92 \keys_define:nn {fontspec-aat}
93 {
94   Vertical .choice: ,
95   Vertical / RotatedGlyphs .code:n =
96     {
97       \__fontspec_update_featstr:n {vertical}
98     }
99 }

```

1.10 Style

```

100 \@@_define_aat_feature_group:n {Style}
101 \@@_define_aat_feature:nnnn {Style} {Italic} {32} {2}
102 \@@_define_aat_feature:nnnn {Style} {Ruby} {28} {2}
103 \@@_define_aat_feature:nnnn {Style} {Display} {19} {1}
104 \@@_define_aat_feature:nnnn {Style} {Engraved} {19} {2}
105 \@@_define_aat_feature:nnnn {Style} {Titling} {19} {4}
106 \@@_define_aat_feature:nnnn {Style} {TitlingCaps} {19} {4} % backwards compat
107 \@@_define_aat_feature:nnnn {Style} {TallCaps} {19} {5}

```

1.11 CJK shape

```

108 \@@_define_aat_feature_group:n {CJKShape}
109 \@@_define_aat_feature:nnnn {CJKShape} {Traditional} {20} {0}
110 \@@_define_aat_feature:nnnn {CJKShape} {Simplified} {20} {1}
111 \@@_define_aat_feature:nnnn {CJKShape} {JIS1978} {20} {2}
112 \@@_define_aat_feature:nnnn {CJKShape} {JIS1983} {20} {3}

```

113 \@@_define_aat_feature:nnnn {CJKShape} {JIS1990} {20} {4}
 114 \@@_define_aat_feature:nnnn {CJKShape} {Expert} {20} {10}
 115 \@@_define_aat_feature:nnnn {CJKShape} {NLC} {20} {13}

1.12 Character width

116 \@@_define_aat_feature_group:n {CharacterWidth}
 117 \@@_define_aat_feature:nnnn {CharacterWidth} {Proportional} {22} {0}
 118 \@@_define_aat_feature:nnnn {CharacterWidth} {Full} {22} {1}
 119 \@@_define_aat_feature:nnnn {CharacterWidth} {Half} {22} {2}
 120 \@@_define_aat_feature:nnnn {CharacterWidth} {Third} {22} {3}
 121 \@@_define_aat_feature:nnnn {CharacterWidth} {Quarter} {22} {4}
 122 \@@_define_aat_feature:nnnn {CharacterWidth} {AlternateProportional} {22} {5}
 123 \@@_define_aat_feature:nnnn {CharacterWidth} {AlternateHalf} {22} {6}
 124 \@@_define_aat_feature:nnnn {CharacterWidth} {Default} {22} {7}

1.13 Annotation

125 \@@_define_aat_feature_group:n {Annotation}
 126 \@@_define_aat_feature:nnnn {Annotation} {Off} {24} {0}
 127 \@@_define_aat_feature:nnnn {Annotation} {Box} {24} {1}
 128 \@@_define_aat_feature:nnnn {Annotation} {RoundedBox} {24} {2}
 129 \@@_define_aat_feature:nnnn {Annotation} {Circle} {24} {3}
 130 \@@_define_aat_feature:nnnn {Annotation} {BlackCircle} {24} {4}
 131 \@@_define_aat_feature:nnnn {Annotation} {Parenthesis} {24} {5}
 132 \@@_define_aat_feature:nnnn {Annotation} {Period} {24} {6}
 133 \@@_define_aat_feature:nnnn {Annotation} {RomanNumerals} {24} {7}
 134 \@@_define_aat_feature:nnnn {Annotation} {Diamond} {24} {8}
 135 \@@_define_aat_feature:nnnn {Annotation} {BlackSquare} {24} {9}
 136 \@@_define_aat_feature:nnnn {Annotation} {BlackRoundSquare} {24} {10}
 137 \@@_define_aat_feature:nnnn {Annotation} {DoubleCircle} {24} {11}

File XVIII

fontspec-code-enc.dtx

1 Extended font encodings

`\EncodingCommand`

```
1 \DeclareDocumentCommand \EncodingCommand { m O{} O{} m }
2 {
3   \bool_if:NF \l_@@_defining_encoding_bool
4     { \@@_error:nn {only-inside-encdef} \EncodingCommand }
5   \DeclareTextCommand{#1}{\UnicodeEncodingName}{#2}[#3]{#4}
6 }
```

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

`\EncodingAccent`

```
7 \DeclareDocumentCommand \EncodingAccent {mm}
8 {
9   \bool_if:NF \l_@@_defining_encoding_bool
10     { \@@_error:nn {only-inside-encdef} \EncodingAccent }
11   \DeclareTextCommand{#1}{\UnicodeEncodingName}{\add@unicode@accent{#2}}
12 }
```

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

`\EncodingSymbol`

```
13 \DeclareDocumentCommand \EncodingSymbol {mm}
14 {
15   \bool_if:NF \l_@@_defining_encoding_bool
16     { \@@_error:nn {only-inside-encdef} \EncodingSymbol }
17   \DeclareTextSymbol{#1}{\UnicodeEncodingName}{#2}
18 }
```

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

`\EncodingComposite`

```
19 \DeclareDocumentCommand \EncodingComposite {mmm}
20 {
21   \bool_if:NF \l_@@_defining_encoding_bool
22     { \@@_error:nn {only-inside-encdef} \EncodingComposite }
23   \DeclareTextComposite{#1}{\UnicodeEncodingName}{#2}{#3}
24 }
```

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

`\EncodingCompositeCommand`

```
25 \DeclareDocumentCommand \EncodingCompositeCommand {mmm}
26 {
27   \bool_if:NF \l_@@_defining_encoding_bool
28     { \@@_error:nn {only-inside-encdef} \EncodingCompositeCommand }
29   \DeclareTextCompositeCommand{#1}{\UnicodeEncodingName}{#2}{#3}
30 }
```


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

`\DeclareUnicodeEncoding`

```
31 \DeclareDocumentCommand \DeclareUnicodeEncoding {mm}
32 {
33   \DeclareFontEncoding{#1}{-}{}
34   \DeclareFontSubstitution{#1}{lrm}{m}{n}
35   \DeclareFontFamily{#1}{lrm}{}
36
37   \DeclareFontShape{#1}{lrm}{m}{n}
38     {<->\UnicodeFontFile{lmroman10-regular}{\UnicodeFontTeXLigatures}}{}
39   \DeclareFontShape{#1}{lrm}{m}{it}
40     {<->\UnicodeFontFile{lmroman10-italic}{\UnicodeFontTeXLigatures}}{}
41   \DeclareFontShape{#1}{lrm}{m}{sc}
42     {<->\UnicodeFontFile{lmromancaps10-regular}{\UnicodeFontTeXLigatures}}{}
43   \DeclareFontShape{#1}{lrm}{bx}{n}
44     {<->\UnicodeFontFile{lmroman10-bold}{\UnicodeFontTeXLigatures}}{}
45   \DeclareFontShape{#1}{lrm}{bx}{it}
46     {<->\UnicodeFontFile{lmroman10-bolditalic}{\UnicodeFontTeXLigatures}}{}
47
48   \tl_set_eq:NN \l_@@_prev_unicode_name_tl \UnicodeEncodingName
49   \tl_set:Nn \UnicodeEncodingName {#1}
50   \bool_set_true:N \l_@@_defining_encoding_bool
51   #2
52   \bool_set_false:N \l_@@_defining_encoding_bool
53   \tl_set_eq:NN \UnicodeEncodingName \l_@@_prev_unicode_name_tl
54 }
```

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

`\UndeclareSymbol` Synonyms for each other but all included for completeness.

```
\UndeclareAccent
\UndeclareCommand
55 \DeclareDocumentCommand \UndeclareSymbol {m}
56 {
57   \bool_if:NF \l_@@_defining_encoding_bool
58     { \@@_error:nn {only-inside-encdef} \UndeclareSymbol }
59   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
60 }
61 \DeclareDocumentCommand \UndeclareAccent {m}
62 {
63   \bool_if:NF \l_@@_defining_encoding_bool
64     { \@@_error:nn {only-inside-encdef} \UndeclareAccent }
65   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
66 }
67 \DeclareDocumentCommand \UndeclareCommand {m}
68 {
69   \bool_if:NF \l_@@_defining_encoding_bool
70     { \@@_error:nn {only-inside-encdef} \UndeclareCommand }
71   \UndeclareTextCommand {#1} {\UnicodeEncodingName}
72 }
```

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

`\UndeclareComposite`

```
73 \DeclareDocumentCommand \UndeclareComposite {mm}
74 {
75   \bool_if:NF \l_@@_defining_encoding_bool
76   { \@@_error:nn {only-inside-encdef} \UndeclareComposite }
77   \cs_undefine:c
78   { \c_backslash_str \UnicodeEncodingName \token_to_str:N #1 - \tl_to_str:n {#2} }
79 }
```

(End 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.

```
1 \ifpackageloaded{euler}
2   { \bool_gset_true:N \g_@@_pkg_euler_loaded_bool }
3   { \bool_gset_false:N \g_@@_pkg_euler_loaded_bool }
4 \cs_new:Nn \fontspec_setup_maths:
5 {
6   \ifpackageloaded{euler}
7     {
8       \bool_if:NTF \g_@@_pkg_euler_loaded_bool
9         { \bool_gset_true:N \g_@@_math_euler_bool }
10        { \@@_error:n {euler-too-late} }
11      }
12    {}
13  \ifpackageloaded{lucbmath}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
14  \ifpackageloaded{lucidabr}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
15  \ifpackageloaded{lucimatx}{ \bool_gset_true:N \g_@@_math_lucida_bool }{}
```

Knuth's CM 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 \TeX '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 `EulerFraktur`, but it's *ugly*. So I ignore it. Sorry if this causes inconvenience.)

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

```

26 \DeclareMathAccent{\dot}      {\mathalpha}{legacymaths}{95}
27 \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{:}\mskip6mu\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}

```

(3A₁₆ = 58₁₀) 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.

```

28 \group_begin:
29 \mathchardef\@tempa="603A \relax
30 \ifx\colon\@tempa
31 \DeclareMathSymbol{\colon}{\mathpunct}{legacymaths}{58}
32 \fi
33 \group_end:

```

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

```

34 \bool_if:NF \g_@@_math_euler_bool
35 {
36 \DeclareMathSymbol{!}{\mathclose}{legacymaths}{33}
37 \DeclareMathSymbol{:}{\mathrel}{legacymaths}{58}
38 \DeclareMathSymbol{;}{\mathpunct}{legacymaths}{59}
39 \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.

```

40 \bool_if:NF \g_@@_math_lucida_bool
41 {
42 \DeclareMathSymbol{0}{\mathalpha}{legacymaths}{`0}
43 \DeclareMathSymbol{1}{\mathalpha}{legacymaths}{`1}
44 \DeclareMathSymbol{2}{\mathalpha}{legacymaths}{`2}

```

```

45 \DeclareMathSymbol{3}{\mathalpha}{legacymaths}{`3}
46 \DeclareMathSymbol{4}{\mathalpha}{legacymaths}{`4}
47 \DeclareMathSymbol{5}{\mathalpha}{legacymaths}{`5}
48 \DeclareMathSymbol{6}{\mathalpha}{legacymaths}{`6}
49 \DeclareMathSymbol{7}{\mathalpha}{legacymaths}{`7}
50 \DeclareMathSymbol{8}{\mathalpha}{legacymaths}{`8}
51 \DeclareMathSymbol{9}{\mathalpha}{legacymaths}{`9}
52 \DeclareMathSymbol{\Gamma}{\mathalpha}{legacymaths}{0}
53 \DeclareMathSymbol{\Delta}{\mathalpha}{legacymaths}{1}
54 \DeclareMathSymbol{\Theta}{\mathalpha}{legacymaths}{2}
55 \DeclareMathSymbol{\Lambda}{\mathalpha}{legacymaths}{3}
56 \DeclareMathSymbol{\Xi}{\mathalpha}{legacymaths}{4}
57 \DeclareMathSymbol{\Pi}{\mathalpha}{legacymaths}{5}
58 \DeclareMathSymbol{\Sigma}{\mathalpha}{legacymaths}{6}
59 \DeclareMathSymbol{\Upsilon}{\mathalpha}{legacymaths}{7}
60 \DeclareMathSymbol{\Phi}{\mathalpha}{legacymaths}{8}
61 \DeclareMathSymbol{\Psi}{\mathalpha}{legacymaths}{9}
62 \DeclareMathSymbol{\Omega}{\mathalpha}{legacymaths}{10}
63 \DeclareMathSymbol{+}{\mathbin}{legacymaths}{43}
64 \DeclareMathSymbol{=}{\mathrel}{legacymaths}{61}
65 \DeclareMathDelimiter{()}{\mathopen}{legacymaths}{40}{largesymbols}{0}
66 \DeclareMathDelimiter{)}{\mathclose}{legacymaths}{41}{largesymbols}{1}
67 \DeclareMathDelimiter{[ ]}{\mathopen}{legacymaths}{91}{largesymbols}{2}
68 \DeclareMathDelimiter{[ ]}{\mathclose}{legacymaths}{93}{largesymbols}{3}
69 \DeclareMathDelimiter{/}{\mathord}{legacymaths}{47}{largesymbols}{14}
70 \DeclareMathSymbol{\mathdollar}{\mathord}{legacymaths}{36}
71 \renewcommand{\hbar}{\mathchar"AF\mkern-9mu h}% TODO: test with other fonts
72 }
73 }

```

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 L^AT_EX 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`.

```

74 \DeclareSymbolFont{operators}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\shapedefault
75 \SetSymbolFont{operators}{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\shapedefault
76 \DeclareSymbolFontAlphabet\mathrm{operators}
77 \SetMathAlphabet\mathit{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\mddefault\itdefault
78 \SetMathAlphabet\mathbf{normal}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\shapedefault
79 \SetMathAlphabet\mathsf{normal}\g_fontspec_encoding_tl\g_@@_mathsf_tl\mddefault\shapedefault
80 \SetMathAlphabet\mathtt{normal}\g_fontspec_encoding_tl\g_@@_mathtt_tl\mddefault\shapedefault
81 \SetSymbolFont{operators}{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\shapedefault
82 \tl_if_empty:NTF \g_@@_bfmathrm_tl
83 {
84   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_mathrm_tl\bfdefault\itdefault
85 }
86 {
87   \SetMathAlphabet\mathrm{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mddefault\shapedefault
88   \SetMathAlphabet\mathbf{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\bfdefault\shapedefault

```

```

89   \SetMathAlphabet\mathit{bold}\g_fontspec_encoding_tl\g_@@_bfmathrm_tl\mdefault\itdefault
90   }
91   \SetMathAlphabet\mathsf{bold}\g_fontspec_encoding_tl\g_@@_mathsf_tl\bfdefault\shapedefault
92   \SetMathAlphabet\mathtt{bold}\g_fontspec_encoding_tl\g_@@_mathtt_tl\bfdefault\shapedefault
93   }

```

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

`\fontspec_maybe_setup_maths`: We're a little less sophisticated about not executing the maths setup if various other maths font packages are loaded. This list is based on the wonderful 'L^AT_EX Font Catalogue': <http://www.tug.dk/FontCatalogue/mathfonts.html>. I'm sure there are more I've missed. Do the T_EX 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.

```

94 \cs_new:Nn \fontspec_maybe_setup_maths:
95 {
96   \@ifpackageloaded{anttor}
97   {
98     \ifx\define@antt@mathversions a\bool_gset_false:N \g_@@_math_bool\fi
99   }{}
100  \@ifpackageloaded{arevmath}      {\bool_gset_false:N \g_@@_math_bool}{}
101  \@ifpackageloaded{eulervm}      {\bool_gset_false:N \g_@@_math_bool}{}
102  \@ifpackageloaded{mathdesign}    {\bool_gset_false:N \g_@@_math_bool}{}
103  \@ifpackageloaded{concmath}     {\bool_gset_false:N \g_@@_math_bool}{}
104  \@ifpackageloaded{cmbright}     {\bool_gset_false:N \g_@@_math_bool}{}
105  \@ifpackageloaded{mathesf}      {\bool_gset_false:N \g_@@_math_bool}{}
106  \@ifpackageloaded{gfsartemisla} {\bool_gset_false:N \g_@@_math_bool}{}
107  \@ifpackageloaded{gfsneohellenic} {\bool_gset_false:N \g_@@_math_bool}{}
108  \@ifpackageloaded{iwona}
109  {
110    \ifx\define@iwona@mathversions a\bool_set_false:N \g_@@_math_bool\fi
111  }{}
112  \@ifpackageloaded{kpfonts}{\bool_gset_false:N \g_@@_math_bool}{}
113  \@ifpackageloaded{kmath}  {\bool_gset_false:N \g_@@_math_bool}{}
114  \@ifpackageloaded{kurier}
115  {
116    \ifx\define@kurier@mathversions a\bool_set_false:N \g_@@_math_bool\fi
117  }{}
118  \@ifpackageloaded{fouriernc}  {\bool_gset_false:N \g_@@_math_bool}{}
119  \@ifpackageloaded{fourier}    {\bool_gset_false:N \g_@@_math_bool}{}
120  \@ifpackageloaded{lmodern}    {\bool_gset_false:N \g_@@_math_bool}{}
121  \@ifpackageloaded{mathpazo}   {\bool_gset_false:N \g_@@_math_bool}{}
122  \@ifpackageloaded{mathptmx}   {\bool_gset_false:N \g_@@_math_bool}{}
123  \@ifpackageloaded{MinionPro}  {\bool_gset_false:N \g_@@_math_bool}{}
124  \@ifpackageloaded{unicode-math} {\bool_gset_false:N \g_@@_math_bool}{}
125  \@ifpackageloaded{breqn}      {\bool_gset_false:N \g_@@_math_bool}{}
126  \@ifpackageloaded{pxfonts}    {\bool_gset_false:N \g_@@_math_bool}{}
127  \@ifpackageloaded{txfonts}    {\bool_gset_false:N \g_@@_math_bool}{}
128  \@ifpackageloaded{newpxmath}  {\bool_gset_false:N \g_@@_math_bool}{}
129  \@ifpackageloaded{newtxmath}  {\bool_gset_false:N \g_@@_math_bool}{}
130  \@ifpackageloaded{mtpro2}     {\bool_gset_false:N \g_@@_math_bool}{}

```

```
131 \bool_if:NT \g_@@_math_bool
132 {
133   \@@_info:n {setup-math}
134   \fontspec_setup_maths:
135 }
136 }
```

```
137 \AtBeginDocument{\fontspec_maybe_setup_maths:}
```

(End 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.

```
1 \bool_if:NT \g_@@_cfg_bool
2   {
3     \InputIfFileExists{fontspec.cfg}
4     {}
5     { \typeout{No~ fontspec.cfg~ file~ found;~ no~ configuration~ loaded.} }
6   }
```


File XXI

fontspec-code-xfss.dtx

1 Changes to the NFSS

```
1 (*fontspec)
```

1.1 Italic small caps and so on

```
2 \providecommand*\scitdefault{\scdefault\itdefault}
```

```
3 \providecommand*\scsldefault{\scdefault\sldefault}
```

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

```
4 \cs_new:Nn \@@_shape_merge:nn { c_@@_shape_#1_#2_tl }
```

```
5 \cs_new:Nn \@@_merge_default_shapes:
```

```
6 {
```

```
7   \tl_const:cn { \@@_shape_merge:nn \itdefault \scdefault } {\scitdefault}
```

```
8   \tl_const:cn { \@@_shape_merge:nn \sldefault \scdefault } {\scsldefault}
```

```
9   \tl_const:cn { \@@_shape_merge:nn \scdefault \itdefault } {\scitdefault}
```

```
10  \tl_const:cn { \@@_shape_merge:nn \scdefault \sldefault } {\scsldefault}
```

```
11  \tl_const:cn { \@@_shape_merge:nn \scsldefault \itdefault } {\scitdefault}
```

```
12  \tl_const:cn { \@@_shape_merge:nn \scitdefault \sldefault } {\scsldefault}
```

```
13  \tl_const:cn { \@@_shape_merge:nn \scitdefault \shapedefault } {\scdefault}
```

```
14  \tl_const:cn { \@@_shape_merge:nn \scsldefault \shapedefault } {\scdefault}
```

```
15 }
```

```
16 \@@_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.

```
17 \prg_new_conditional:Nnn \@@_if_merge_shape:n {TF}
```

```
18 {
```

```
19   \bool_lazy_and:nnTF
```

```
20     { \tl_if_exist_p:c { \@@_shape_merge:nn {\f@shape} {#1} } }
```

```
21     {
```

```
22       \cs_if_exist_p:c
```

```
23         {
```

```
24           \f@encoding/\f@family/\f@series/
```

```
25           \tl_use:c { \@@_shape_merge:nn {\f@shape} {#1} }
```

```
26         }
```

```
27       }
```

```
28     \prg_return_true: \prg_return_false:
```

```
29   }
```

```
30 \cs_if_exist:NTF \DeclareFontSeriesChangeRule
```

```
31 {
```

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

```
33 }
```

```
34 {
```

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

writing `\textit{\textsc{...}}` instead.

```
35 \DeclareRobustCommand{\sishape}
36 {
37   \not@math@alphabet\sishape\relax
38   \fontshape{\scitdefault}\selectfont
39 }
40 \DeclareTextFontCommand{\textsi}{\sishape}
```

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

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

```
41 \cs_new:Nn \@@_merge_shape:n
42 {
43   \@@_if_merge_shape:nTF {#1}
44     { \fontshape { \tl_use:c { \@@_shape_merge:nn {f@shape} {#1} } } \selectfont }
45     { \fontshape {#1} \selectfont }
46 }
```

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

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

```
\scshape 47 \DeclareRobustCommand \itshape
\upshape 48 {
\slshape 49   \not@math@alphabet\itshape\mathit
50   \@@_merge_shape:n\itdefault
51 }
52 \DeclareRobustCommand \slshape
53 {
54   \not@math@alphabet\slshape\relax
55   \@@_merge_shape:n\sldefault
56 }
57 \DeclareRobustCommand \scshape
58 {
59   \not@math@alphabet\scshape\relax
60   \@@_merge_shape:n\scdefault
61 }
62 \DeclareRobustCommand \upshape
63 {
64   \not@math@alphabet\upshape\relax
65   \@@_merge_shape:n\updefault
66 }
```

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

1.2 Emphasis

`\emfontdeclare`

```
67 \cs_set_protected:Npn \emfontdeclare #1
68 {
69   \prop_gclear:N \g_@@_em_prop
70   \int_zero:N \l_@@_emdef_int
```

```

71 \bool_gset_true:N \g_@@_em_normalise_slant_bool
72
73 \tl_if_in:nnT {#1} {\slshape}
74 {
75   \tl_if_in:nnT {#1} {\itshape}
76   {
77     \bool_gset_false:N \g_@@_em_normalise_slant_bool
78   }
79 }
80
81 \group_begin:
82   \normalfont
83   \clist_map_inline:nn {\emreset,#1}
84   {
85     ##1
86     \prop_gput_if_new:NxV \g_@@_em_prop { \f@shape } { \l_@@_emdef_int }
87     \prop_gput:Nxn \g_@@_em_prop { switch-\int_use:N \l_@@_emdef_int } { ##1 }
88     \int_incr:N \l_@@_emdef_int
89   }
90 \group_end:
91 }

```

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

`\em`

```

92 \DeclareRobustCommand \em
93 {
94   \@nomath\em
95   \tl_set:Nx \l_@@_emshape_query_tl { \f@shape }
96
97   \bool_if:NT \g_@@_em_normalise_slant_bool
98   {
99     \tl_replace_all:Nnn \l_@@_emshape_query_tl {/sl} {/it}
100   }
101
102 <debug> \typeout{Emph~ level:~\int_use:N \l_@@_em_int}
103   \prop_get:NxNT \g_@@_em_prop { \l_@@_emshape_query_tl } \l_@@_em_tmp_tl
104   {
105     \int_set:Nn \l_@@_em_int { \l_@@_em_tmp_tl }
106 <debug> \typeout{Shape~ (\l_@@_emshape_query_tl)~ detected;~ new~ level:~\int_use:N \l_@@_em_
107   }
108
109   \int_incr:N \l_@@_em_int
110
111   \prop_get:NxNTF \g_@@_em_prop { switch-\int_use:N \l_@@_em_int } \l_@@_em_switch_tl
112   { \l_@@_em_switch_tl }
113   {
114     \int_zero:N \l_@@_em_int
115     \emreset
116   }
117
118 }

```

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

```
\emph
\emshape 119 \DeclareTextFontCommand{\emph}{\em}
\emminnershape 120 \cs_set:Npn \emreset { \upshape }
\emreset 121 \cs_set:Npn \emshape { \itshape }
122 \cs_set:Npn \emminnershape { \upshape }
```

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

```
123 \emfontdeclare{ \emshape, \emminnershape }
124 \cs_new:Nn \fontspec_set_em_level:n { \int_set:Nn \l_@@_em_int {#1} }
125 }
```

1.3 Strong emphasis

`\strongfontdeclare`

```
126 \cs_set_protected:Npn \strongfontdeclare #1
127 {
128   \prop_gclear:N \g_@@_strong_prop
129   \int_zero:N \l_@@_strongdef_int
130
131   \group_begin:
132     \normalfont
133     \clist_map_inline:nn {\strongreset,#1}
134     {
135       ##1
136       \prop_gput_if_new:NxV \g_@@_strong_prop { \f@series } { \l_@@_strongdef_int }
137       \prop_gput:Nxn \g_@@_strong_prop { switch-\int_use:N \l_@@_strongdef_int } { ##1 }
138       \int_incr:N \l_@@_strongdef_int
139     }
140   \group_end:
141 }
```

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

`\strongenv`

```
142 \DeclareRobustCommand \strongenv
143 {
144   \@nomath\strongenv
145
146   <debug> \typeout{Strong~ level:-\int_use:N \l_@@_strong_int}
147   \prop_get:NxNT \g_@@_strong_prop { \f@series } \l_@@_strong_tmp_tl
148   {
149     \int_set:Nn \l_@@_strong_int { \l_@@_strong_tmp_tl }
150   <debug> \typeout{Series~ (\f@series)~ detected;~ new~ level:-\int_use:N \l_@@_strong_int}
151   }
152
153   \int_incr:N \l_@@_strong_int
154
155   \prop_get:NxNTF \g_@@_strong_prop { switch-\int_use:N \l_@@_strong_int } \l_@@_strong_sw
156   { \l_@@_strong_switch_tl }
```

```

157     {
158         \int_zero:N \l_@@_strong_int
159         \strongreset
160     }
161 }
162 }

```

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

```

\strong
\strongreset 163 \DeclareTextFontCommand{\strong}{\strongenv}
164 \cs_set:Npn \strongreset {}

```

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

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

```

165 \cs_set:Npn \reset@font
166     {
167         \normalfont
168         \int_zero:N \l_@@_em_int
169         \int_zero:N \l_@@_strong_int
170     }

```

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

Programmer's interface for setting nesting levels:

```

171 \cs_new:Nn \fontspec_set_strong_level:n { \int_set:Nn \l_@@_strong_int {#1} }

```

Defaults:

```

172 \strongfontdeclare{ \bfseries }
173 </fontspec>

```

Index

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

Symbols	
\#	250, 251, 282
\,	1, 2, 3, 4, 5, 17
\-	685
@@ commands:	
\@@_DeclareFontShape:nnnnn	501, 508, <u>518</u> , 536, 549, 557, 570
\g_@@_OT_features_prop	9, 11, 71
\@@_add_nfssfont:nnnn	291, 305, 306, 307, 308, 309, 310, <u>324</u>
\@@_aff_error:n	11, 352, 393, 425
\l_@@_alias_bool	22, 206, 213, 219, 224, 231, 251
\l_@@_all_features_clist	21, 53, 99, 109, 123, 181, 271
\g_@@_all_keyval_modules_clist	1, 47, 208, 226
\g_@@_all_opentype_feature_names_prop	72, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343
\l_@@_arg_clist	59, 278, 279, 280, 283, 286
\l_@@_atsui_bool	7, 10, 204, 343, 352, 642
\l_@@_basename_tl	10, 42, 88, 321
\l_@@_bf_series_seq	45, 148, 160, 163
\g_@@_bfmathrm_tl	71, 72, 82, 87, 88, 89, 119
\@@_calc_scale:n	298, 299, <u>309</u>
\g_@@_cfg_bool	1, 7, 8, 17
\l_@@_check_bool	5, 65, 66, 195, 200, 206, 217
\@@_check_lang:Nn	121
\@@_check_lang:NnTF	97, <u>121</u> , 337
\@@_check_lang:Nnn	125
\@@_check_lang:NnnTF	110, <u>121</u>
\@@_check_ot_feat:Nn	169
\@@_check_ot_feat:NnTF	50, 60, <u>169</u> , 634
\@@_check_ot_feat:Nnnn	174
\@@_check_ot_feat:NnnnTF	67, <u>169</u>
\@@_check_script:Nn	75
\@@_check_script:NnTF	75, 80, 291
\@@_combo_sc_shape:n	509, 512, 558, 595, 603
\@@_construct_font_call:mn	136, 138, 142, 145, <u>151</u> , 176, 274, 390, 391, 412, 495
\@@_construct_font_call:nnnnn	151, 160
\l_@@_curr_bfname_tl	91, 158, 168, 171, 173, 205
\l_@@_curr_fontname_tl	90, 315, 316
\g_@@_curr_series_tl	89, 147, 162, 166, 171, 173, 205, 681
\@@_declare_shape:nnnn	401, <u>420</u>
\@@_declare_shape_loginfo:nn	436, <u>578</u>
\@@_declare_shape_slanted:nn	434, <u>528</u>
\@@_declare_shapes_bx:nn	435, <u>540</u>
\@@_declare_shapes_normal:nn	432, <u>499</u>
\@@_declare_shapes_smcaps:nn	433, <u>504</u>
\g_@@_default_fontopts_clist	46, 111, 121
\@@_define_aat_feature:nnnn	2, 3, 4, <u>5</u> , 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58, 60, 61, 62, 63, 65, 66, 67, 101, 102, 103, 104, 105, 106, 107, 109, 110, 111, 112, 113, 114, 115, 117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 185

<p> \@@_define_aat_feature_group:n 1, 1, 29, 35, 44, 55, 59, 64, 68, 80, 91, 100, 108, 116, 125, 180 \@@_define_opentype_feature:nnnnn 5, 7, 29, 42, 43, 44, 45, 49, 50, 61, 77, 93, 105, 111, 112, 113, 115, 120, 121, 122, 125, 126, 127, 129, 153, 154, 157, 177, 195 \@@_define_opentype_feature_- group:n 1, 6, 28, 41, 60, 76, 92, 104, 114, 124, 128, 156, 176, 190, 194, 203, 222, 236, 258, 268 \@@_define_opentype_onoffreset:nnnnn 13, 14, 15, 16, 17, 18, 27, 34, 35, 36, 37, 38, 39, 39, 40, 51, 52, 53, 54, 55, 59, 70, 71, 72, 73, 74, 75, 86, 87, 88, 89, 90, 91, 100, 101, 102, 103, 110, 123, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 155, 168, 169, 170, 171, 172, 173, 174, 175, 187, 188, 189, 190, 191, 192, 193, 195, 196, 197, 198, 199, 200, 201, 202 \@@_define_opentype_onreset:nnnnn 47 \g_@@_defined_shapes_tl 86, 184, 580, 680 \l_@@_defining_encoding_bool .. 3, 9, 15, 21, 24, 27, 50, 52, 57, 63, 69, 75 \l_@@_disable_defaults_bool 21, 97, 155 \l_@@_em_int 36, 102, 105, 106, 109, 111, 114, 124, 168 \g_@@_em_normalise_slant_bool 26, 71, 77, 97 \g_@@_em_prop .. 69, 73, 86, 87, 103, 111 \l_@@_em_switch_tl 111, 111, 112 \l_@@_em_tmp_tl 103, 105, 116 \l_@@_emdef_int 37, 70, 86, 87, 88 \l_@@_emshape_query_tl 95, 99, 103, 106, 110 \@@_error:n 1, 10, 446 \@@_error:nn 2, 3, 4, 10, 14, 16, 22, 28, 58, 64, 70, 76, 140, 413, 716, 735 \@@_error:nnn 421 \g_@@_euenc_bool 9, 10, 19, 23, 36, 39, 58 \l_@@_ext_filename_tl 85, 86, 89, 90, 92 \l_@@_extension_tl 39, 45, 53, 72, 93, 116, 162 \l_@@_extensions_clist 48, 50, 61, 256 \l_@@_external_bool ... 23, 28, 40, 374 \@@_extract_all_features: 94 \@@_extract_all_features:n ... 20, 94 </p>	<p> \l_@@_fake_embolden_tl 127, 561, 564, 578 \l_@@_fake_slant_tl . 126, 556, 583, 586 \l_@@_family_fontopts_clist 52, 105, 106, 112 \g_@@_family_int_prop ... 76, 248, 254 \l_@@_family_label_tl 105, 107, 125, 150, 164 \@@_feat_off:n 39, 44 \@@_feat_prop_add:nn . 1, 2, 3, 4, 5, 17 \@@_feat_reset:n 40, 45, 50 \@@_find_autofonts: 262, 282 \l_@@_firsttime_bool 1, 29, 179, 225, 273, 349, 439, 450, 462, 516, 554, 576, 653, 674 \@@_font_is_file: 30, 168, 175 \@@_font_is_name: 168, 675 \l_@@_font_path_tl ... 29, 94, 174, 676 \@@_font_suppress_not_found_- error: 5, 9, 38, 269 \l_@@_fontcfg_bool ... 12, 13, 18, 22, 81 \l_@@_fontface_cs_tl 17, 60, 144, 148, 149, 154, 159, 160, 163, 167, 291, 318, 337, 411, 416, 634, 645 \l_@@_fontfeat_bf_clist 62, 202, 306, 579 \l_@@_fontfeat_bfit_clist 64, 213, 309, 563, 565, 585, 587 \l_@@_fontfeat_bfsl_clist 66, 221, 310 \l_@@_fontfeat_clist 57, 130, 193, 274 \l_@@_fontfeat_curr_clist 58, 455, 464, 477 \l_@@_fontfeat_it_clist 63, 209, 307, 557 \l_@@_fontfeat_sc_clist . 67, 227, 455 \l_@@_fontfeat_sl_clist . 65, 217, 308 \l_@@_fontfeat_up_clist 61, 198, 233, 305 \g_@@_fontid_family_prop 75, 228, 256 \l_@@_fontid_tl 21, 23, 42, 95, 224, 228, 236 \l_@@_fontname_bf_tl 75, 129, 168, 287, 293, 306, 580 \l_@@_fontname_bfit_tl 76, 131, 179, 286, 287, 288, 309, 566, 588 \l_@@_fontname_bfsl_tl 133, 183, 301, 310 \l_@@_fontname_it_tl 74, 130, 138, 286, 298, 307, 558 \l_@@_fontname_sc_tl 134, 193, 459, 471 \l_@@_fontname_sl_tl 132, 143, 301, 308 </p>
--	---

\l_@@_fontname_tl	96, 154, 156, 160	\l_@@_lang_name_tl	112, 131, 139, 140, 183, 185
\l_@@_fontname_up_tl	9, 42, 45, 126, 128, 128, 136, 138, 140, 142, 145	\l_@@_lang_tl	48, 138, 171, 280, 339, 356, 619, 628
\@@_fontname_wrap:n	47, 153, 154, 170, 174	\l_@@_language_int	30, 45, 187, 188, 192, 198, 278, 340, 357
\l_@@_fontopts_clist	51, 102, 103, 113, 401, 408, 409, 410	\l_@@_leftover_clist	54, 399, 401
\g_@@_fontopts_prop	68, 87, 102, 105, 134, 137, 141, 142, 408	\@@_load_external_fontoptions:Nn	18, 79, 407
\@@_get_features:Nn	60, 189	\@@_load_font:	26, 132
\@@_get_features:n	28, 189, 275, 484	\@@_load_fontname:Nn 400, 404, 450, 471	
\l_@@_graphite_bool	11, 204, 346, 364	\@@_lua_function:n	78
\l_@@_harfbuzz_bool	10, 74, 97	\@@_lua_function:nnn	78
\c_@@_hexcol_tl	150, 217, 696	\@@_lua_function:nnnn	78, 163
\l_@@_hexcol_tl	143, 216, 218, 430, 434, 447, 696	\@@_lua_function:nnnnn	78, 216
\l_@@_hyphenchar_tl	142, 413, 414, 416, 419	\@@_main_DeclareFontExtensions:n	122, 254
\@@_if_autofont:nn	387	\@@_main_IfFontFeatureActiveTF:nnn	126, 259
\@@_if_autofont:nnTF	380	\@@_main_addfontfeatures:n 82, 86, 146	
\@@_if_detect_external:n	58	\@@_main_aliasfontfeature:nn 106, 203	
\@@_if_detect_external:nTF 12, 58, 175		\@@_main_aliasfontfeatureoption:nnn	110, 222
\@@_if_font_feature:n	265	\@@_main_fontspec:nn	1, 3
\@@_if_font_feature:nTF	263	\@@_main_liningnums:n	137, 294
\@@_if_merge_shape:n	17	\@@_main_newAATfeature:nnnn	94, 177
\@@_if_merge_shape:nTF	43, 191	\@@_main_newfontface:NnnN	59, 63, 67, 71, 114
\@@_info:n	7, 133, 307, 326	\@@_main_newfontfamily:NnnN	43, 47, 51, 55, 101, 116
\@@_info:nn	8, 382	\@@_main_newfontfeature:nn	90, 170
\@@_info:nnn	9, 265	\@@_main_newopentypefeature:nnn	98, 102, 187
\@@_init:	6, 174, 270, 670	\@@_main_oldstylenums:n	132, 287
\@@_init_fontface:	192, 691	\@@_main_setboldmathrm:nn	27, 69
\@@_init_ttc:n	17, 70	\@@_main_setmainfont:nn	8, 24, 39
\@@_int_mult_truncate:Nn	74, 459	\@@_main_setmathrm:nn	23, 63
\@@_iv_str_to_num:Nn	86, 135, 136, 184, 188, 189, 719	\@@_main_setmathsf:nn	31, 75
\@@_iv_str_to_num:w	729, 730, 732	\@@_main_setmathtt:nn	35, 81
\@@_keys_define_code:nnn	7, 13, 16, 20, 24, 36, 37, 46, 104, 109, 114, 121, 126, 130, 141, 145, 150, 177, 181, 185, 196, 200, 207, 211, 215, 219, 223, 230, 235, 241, 245, 249, 253, 257, 261, 262, 263, 264, 265, 266, 267, 271, 275, 294, 303, 347, 354, 373, 394, 398, 402, 426, 456, 472, 476, 482, 493, 497, 501, 594, 598	\@@_main_setmonofont:nn	18, 50
\l_@@_keys_leftover_clist	55, 124, 127, 128, 129, 194, 195, 199, 200, 203, 205, 209, 210	\@@_main_setsansfont:nn	13, 37
\@@_keys_set_known:nnN	67, 122, 127, 129, 193, 195, 330, 399	\@@_make_AAT_feature:nn	9, 12, 76, 87
		\@@_make_AAT_feature_string:Nnn	26
		\@@_make_AAT_feature_string:NnnTF	12, 17, 26, 644
		\@@_make_OT_feature:nnn	33, 52, 71, 211, 218, 231, 242, 265, 275
		\@@_make_font_shapes:Nnnnn	316, 396
		\@@_make_ot_smallcaps:TF	631
		\@@_make_smallcaps:TF	461, 631, 637

<code>\l_@@_mapping_tl</code>	<code>\l_@@_optical_size_tl</code>
. 22, 23, 24, 26, 146, 213, 214, 474, 478 145, 165, 506, 523, 677
<code>\g_@@_math_bool</code>	<code>\l_@@_options_tl</code> ... 97, 153, 156, 160
... 5, 6, 18, 98, 100, 101, 102, 103,	<code>\l_@@_ot_bool</code>
104, 105, 106, 107, 110, 112, 113,	8, 28, 39,
116, 118, 119, 120, 121, 122, 123,	65, 78, 91, 108, 121, 136, 197, 271,
124, 125, 126, 127, 128, 129, 130, 131	344, 360, 369, 504, 514, 616, 639, 673
<code>\g_@@_math_euler_bool</code>	<code>\@@_ot_compat:nn</code> ... 360, 364, 365,
9, 14, 34	366, 367, 368, 369, 370, 371, 372,
<code>\g_@@_math_lucida_bool</code>	373, 374, 375, 376, 377, 378, 379, 380
..... 13, 14, 15, 15, 40	<code>\@@_ot_validate_tag:n</code>
<code>\g_@@_mathrm_tl</code> 102, 158, 159, 210, 211, 212, 701
65,	<code>\@@_ot_validate_tag:w</code>
66, 74, 75, 77, 78, 81, 84, 98, 118, 122	704, 707
<code>\g_@@_mathsf_tl</code>	<code>\@@_ot_validate_tag_aux:w</code> 710, 711, 713
..... 77, 78, 79, 91, 99, 120, 123	<code>\g_@@_pkg_euler_loaded_bool</code> 2, 3, 8, 16
<code>\g_@@_mathtt_tl</code>	<code>\c_@@_postadjust_tl</code>
..... 80, 83, 84, 92, 100, 121, 124	152, 697
<code>\@@_merge_default_shapes:</code> ... 5, 16	<code>\l_@@_postadjust_tl</code>
<code>\@@_merge_shape:n</code> ... 41, 50, 55, 60, 65 149, 396, 406, 418, 502,
<code>\l_@@_mm_bool</code> ... 9, 345, 356, 509, 514	509, 537, 552, 560, 573, 606, 609, 697
<code>\l_@@_mode_tl</code> 76, 84, 86, 90, 99, 625, 684	<code>\l_@@_pre_feat_sclist</code>
<code>\@@_msg_new:nnn</code>	138,
. 13, 18, 23, 44, 84, 90, 94, 104, 108,	145, 154, 275, 390, 391, 412, 496, 613
112, 117, 122, 127, 133, 137, 143,	<code>\@@_preparse_features:</code>
147, 151, 156, 160, 165, 169, 174,	25, 118
178, 186, 190, 194, 198, 202, 207, 212	<code>\l_@@_prev_unicode_name_tl</code> . 48, 53, 99
<code>\@@_msg_new:nnnn</code> 15, 28, 37, 48, 58, 66, 74	<code>\l_@@_primitive_font</code>
<code>\l_@@_never_check_bool</code>	39, 40
..... 28, 78, 128, 176, 272	<code>\@@_primitive_font_current_name:</code>
<code>\g_@@_nfss_enc_tl</code> 56, 184, 186
4,	<code>\@@_primitive_font_get_name:N</code> 56, 392
31, 33, 44, 46, 57, 59, 100, 109, 263,	<code>\@@_primitive_font_glyph_if_</code>
269, 501, 508, 536, 549, 557, 570, 682	exist:Nn
<code>\l_@@_nfss_fam_tl</code> .. 104, 226, 241, 273	44
<code>\g_@@_nfss_family_tl</code>	<code>\@@_primitive_font_glyph_if_</code>
..... 41, 101, 163, 180, 230, 241,	exist:NnTF
242, 255, 256, 263, 269, 270, 271,	44, 416
272, 277, 278, 279, 280, 501, 508,	<code>\@@_primitive_font_gset:Nnn</code> ...
536, 537, 549, 551, 557, 559, 570, 572 1, 26, 28, 33
<code>\l_@@_nfss_prop</code>	<code>\@@_primitive_font_gset:NnnTF</code> ... 34
69, 171, 204	<code>\@@_primitive_font_gset:NnnTTF</code> . 21
<code>\l_@@_nfss_sc_tl</code>	<code>\@@_primitive_font_gset:Onn</code> .. 33, 144
. 102, 424, 430, 476, 506, 509, 555, 605	<code>\@@_primitive_font_gset:OnnTF</code> ... 34
<code>\l_@@_nfss_tl</code> 103, 423, 429, 451, 502, 593	<code>\@@_primitive_font_gset:p:NnnTF</code> . 21
<code>\l_@@_nfssfont_prop</code>	<code>\@@_primitive_font_if_exist:nTF</code>
70, 311, 334 35, 176
<code>\l_@@_nobf_bool</code>	<code>\@@_primitive_font_if_null:NnTF</code> .
..... 2, 26, 154, 157, 284, 291, 581 13, 24, 29, 40
<code>\l_@@_noit_bool</code>	<code>\@@_primitive_font_if_null_p:N</code> .. 13
..... 3, 27, 134, 137, 284, 296, 559	<code>\@@_primitive_font_set:Nnn</code>
<code>\l_@@_nosc_bool</code> 4, 189, 192, 457, 468, 474 1, 21, 23, 31, 39, 390, 391
<code>\c_@@_opacity_tl</code> 151, 217, 448, 460, 695	<code>\@@_primitive_font_set:NnnTF</code> . 32, 137
<code>\l_@@_opacity_tl</code>	<code>\@@_primitive_font_set:NnnTTF</code> .. 21
. 144, 216, 218, 448, 453, 460, 465, 695	<code>\@@_primitive_font_set:Onn</code>
	31
	<code>\@@_primitive_font_set:OnnTF</code> . 32, 411
	<code>\@@_primitive_font_set_hyphenchar:Nn</code>
 52, 407, 419
	<code>\@@_primitive_font_set_p:NnnTF</code> .. 21

<code>\l_@@_proceed_bool</code>	27, 56, 63, 69
<code>\l_@@_punctspace_adjust_tl</code>	
	147, 152, 379, 384, 389, 699
<code>\g_@@_rawfeatures_sclist</code>	59,
		153, 275, 278, 485, 496, 657, 666, 693
<code>\@@_remove_clashing_featstr:n</code>	..	
	23, 67, 660
<code>\l_@@_renderer_tl</code>	52,
		55, 59, 60, 83, 164, 353, 361, 365, 679
<code>\l_@@_rmfamily_encoding_tl</code>	
	9, 15, 21, 31, 158
<code>\l_@@_rmfamily_family_tl</code>	..	29, 30, 155
<code>\@@_sanitise_fontname:Nn</code>	
	8, 9, 10, 44, 74, 75, 76, 84, 130
<code>\@@_save_family:nn</code>	33, 259
<code>\@@_save_family_needed:n</code>	220
<code>\@@_save_family_needed:nTF</code>	...	31, 220
<code>\@@_save_fontid_family:nn</code>	236, 246, 258	
<code>\@@_save_fontinfo:n</code>	261, 267
<code>\l_@@_saved_fontname_tl</code>	..	98, 425, 442
<code>\l_@@_scale_tl</code>	141, 192, 301,
		305, 306, 320, 329, 487, 489, 494, 694
<code>\l_@@_script_int</code>	29, 42,
		94, 136, 138, 144, 189, 192, 198, 277, 296
<code>\l_@@_script_name_tl</code>	107, 131,
		136, 137, 141, 181, 183, 184, 294, 308
<code>\l_@@_script_tl</code>	
		47, 95, 123, 135, 171, 279, 295, 618, 627
<code>\l_@@_scriptlang_exist_bool</code>	
	25, 288, 297, 303, 334, 342, 346
<code>\@@_select_font_family:nn</code>	
	1, 43, 151, 156, 159, 165
<code>\@@_set_autofont:Nnn</code>	
	286, 287, 288, 293, 298, 301, 372
<code>\@@_set_default_features:nn</code>	..	76, 118
<code>\@@_set_faces:</code>	264, 303
<code>\@@_set_faces_aux:nnnnn</code>	311, 313
<code>\@@_set_family:NnnN</code>	147, 156, 157
<code>\@@_set_font_default_features:nnn</code>	
	77, 123
<code>\@@_set_font_dimen:NnN</code>	..	317, 318, 331
<code>\@@_set_font_type:N</code>	9,
		27, 38, 64, 77, 90, 107, 120, 135, 143, 338
<code>\@@_set_fontface:NNnnN</code>	..	161, 169, 170
<code>\@@_set_scriptlang:</code>	27, 176
<code>\@@_setboldmathrm_hook:nn</code>	...	73, 93
<code>\@@_setmainfont_hook:nn</code>	34, 87
<code>\@@_setmathrm_hook:nn</code>	67, 90
<code>\@@_setmathsf_hook:nn</code>	79, 91
<code>\@@_setmathtt_hook:nn</code>	85, 92
<code>\@@_setmonofont_hook:nn</code>	60, 89
<code>\@@_setsansfont_hook:nn</code>	47, 88
<code>\@@_setup_nfss:Nnnn</code>	451, 476, 480
<code>\@@_setup_single_size:nn</code>	...	427, 439
<code>\l_@@_sffamily_encoding_tl</code>	
	10, 17, 22, 44, 159
<code>\l_@@_sffamily_family_tl</code>	..	42, 43, 156
<code>\@@_shape_merge:nn</code>	...	4, 7, 8, 9, 10,
		11, 12, 13, 14, 20, 25, 44, 193, 514, 515
<code>\l_@@_shaper_tl</code>	81, 85, 86, 91, 100, 626	
<code>\g_@@_single_feat_tl</code>	60, 87,
		88, 267, 279, 281, 283, 298, 341, 358, 655
<code>\l_@@_size_tl</code>	
	105, 255, 441, 446, 447, 482, 494
<code>\l_@@_sizedfont_tl</code>	
	106, 259, 442, 450, 452
<code>\l_@@_sizefeat_clist</code>	
	48, 49, 232, 237, 329, 335
<code>\l_@@_sizing_leftover_clist</code>	
	56, 445, 451, 477
<code>\l_@@_smcp_shape_tl</code>	
	109, 193, 196, 199, 202
<code>\@@_strip_leading_sign:Nw</code>	..	723, 726
<code>\@@_strip_plus_minus:n</code>	196, 198
<code>\@@_strip_plus_minus_aux:Nq</code>	..	198, 199
<code>\l_@@_strnum_int</code>	
		31, 86, 92, 135, 146, 184, 199, 296, 340
<code>\l_@@_strong_int</code>	38,
		146, 149, 150, 153, 155, 158, 169, 171
<code>\g_@@_strong_prop</code>	
	74, 128, 136, 137, 147, 155
<code>\l_@@_strong_switch_tl</code>	..	112, 155, 156
<code>\l_@@_strong_tmp_tl</code>	117, 147, 149
<code>\l_@@_strongdef_int</code>	
	39, 129, 136, 137, 138
<code>\@@_swap_plus_minus:n</code>	67, 72
<code>\@@_swap_plus_minus_aux:Nq</code>	...	72, 73
<code>\l_@@_test_font</code>	137, 143
<code>\l_@@_tfm_bool</code>	6, 342, 349
<code>\l_@@_this_feat_clist</code>	60, 279, 287, 292	
<code>\l_@@_this_font_tl</code>	107, 238,
		239, 243, 277, 286, 292, 326, 332, 335
<code>\@@_tl_new_if_free:N</code>	146, 152
<code>\l_@@_tmp_int</code>	...	32, 458, 459, 467, 468
<code>\l_@@_tmp_tl</code>	41, 42, 44,
		45, 93, 94, 104, 105, 106, 107, 113,
		123, 124, 129, 130, 134, 137, 138,
		139, 141, 142, 152, 161, 162, 163,
		214, 215, 216, 228, 230, 234, 235,
		236, 248, 250, 251, 253, 254, 255, 330
<code>\l_@@_tmpa_bool</code>	20, 63, 66, 68

<code>\clist_map_inline:n</code>	74, 83, 86, 126, 133, 210, 229, 250, 289, 335, 427, 663
<code>\clist_new:N</code>	46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67
<code>\clist_put_left:Nn</code>	464
<code>\clist_put_right:Nn</code>	198, 202, 209, 213, 217, 221, 227, 233, 557, 563, 565, 579, 585, 587
<code>\clist_set:Nn</code>	49, 99, 109, 232, 237, 256, 278, 329
<code>\clist_set_eq:NN</code>	279, 455
<code>\colon</code>	30, 31, 116
<code>\convertcolorspec</code>	430
cs commands:	
<code>\cs:w</code>	129
<code>\cs_end:</code>	129
<code>\cs_generate_variant:Nn</code>	11, 12, 71, 73, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 158, 258, 438, 527, 669, 706, 725
<code>\cs_if_eq:NNTF</code>	103, 160, 213
<code>\cs_if_exist:NTF</code>	3, 7, 27, 30, 199, 428
<code>\cs_if_exist_p:N</code>	22
<code>\cs_new:Nn</code>	1, 1, 1, 4, 4, 5, 5, 5, 7, 11, 12, 13, 15, 15, 24, 37, 39, 40, 41, 41, 44, 47, 50, 52, 52, 63, 67, 69, 70, 72, 74, 75, 79, 81, 94, 94, 101, 114, 118, 118, 123, 124, 132, 146, 146, 147, 151, 156, 157, 158, 161, 168, 169, 170, 170, 171, 172, 176, 177, 187, 189, 198, 203, 222, 246, 254, 259, 259, 267, 282, 283, 303, 309, 313, 318, 324, 330, 331, 338, 360, 372, 396, 404, 420, 439, 480, 499, 504, 512, 518, 528, 540, 578, 631, 632, 637, 650, 660, 691, 720
<code>\cs_new:Npn</code>	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 57, 65, 66, 73, 199, 239
<code>\cs_new_eq:NN</code>	56
<code>\cs_new_protected:Nn</code>	1, 287, 294, 702
<code>\cs_set:Npn</code>	1, 5, 9, 31, 32, 33, 34, 61, 79, 80, 81, 82, 120, 121, 122, 164, 165, 174, 353, 445, 670, 707, 713, 726, 732
<code>\cs_set_eq:NN</code>	32, 43, 60, 63, 87, 88, 89, 90, 91, 92, 93, 170, 180
<code>\cs_set_protected:Npn</code>	67, 126
<code>\cs_to_str:N</code>	103, 105, 108, 129, 162, 215
<code>\cs_undefine:N</code>	77, 242, 523
<code>\cyrillicencoding</code>	51, 55, 81
	D
<code>\date</code>	55
<code>\ddot</code>	20
<code>\DeclareDocumentCommand</code>	1, 7, 13, 19, 25, 31, 51, 55, 61, 67, 67, 73
<code>\DeclareEmphSequence</code>	32
<code>\DeclareFontEncoding</code>	30, 33
<code>\DeclareFontExtensions</code>	120
<code>\DeclareFontFamily</code>	35, 263
<code>\DeclareFontSeriesChangeRule</code>	30
<code>\DeclareFontSeriesDefault</code>	26, 27, 39, 40, 52, 53
<code>\DeclareFontShape</code>	37, 39, 41, 43, 45, 53, 54, 525
<code>\DeclareFontSubstitution</code>	31, 34
<code>\DeclareMathAccent</code>	18, 19, 20, 21, 22, 23, 24, 25, 26, 27
<code>\DeclareMathDelimiter</code>	65, 66, 67, 68, 69
<code>\DeclareMathSymbol</code>	31, 36, 37, 38, 39, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 70
<code>\DeclareOption</code>	1, 5, 6, 7, 8, 9, 10, 11, 16
<code>\DeclareRobustCommand</code>	35, 47, 52, 57, 62, 92, 142
<code>\DeclareSymbolFont</code>	16, 74
<code>\DeclareSymbolFontAlphabet</code>	76
<code>\DeclareTextCommand</code>	5, 11
<code>\DeclareTextComposite</code>	23
<code>\DeclareTextCompositeCommand</code>	29
<code>\DeclareTextFontCommand</code>	40, 119, 163
<code>\DeclareTextSymbol</code>	17
<code>\DeclareUnicodeEncoding</code>	21, 31
<code>\def</code>	27
<code>\defaultfontfeatures</code>	73
<code>\Delta</code>	53
dim commands:	
<code>\dim_compare:nNnTF</code>	334
<code>\dim_eval:n</code>	3, 7
<code>\dim_new:N</code>	42, 43, 44
<code>\dim_set:Nn</code>	333
<code>\dim_to_fp:n</code>	322, 323
dim internal commands:	
<code>__dim_eval:w</code>	76
<code>__dim_eval_end:</code>	76
<code>\dot</code>	26
<code>\DTX</code>	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23
	E
<code>\else</code>	17, 95, 202, 741, 742

`\fontspec_if_language:n` 86
`\fontspec_if_language:nn` 103
`\fontspec_if_language:nnTF` 103
`\fontspec_if_language:nTF` 86
`\fontspec_if_opentype:` 23
`\fontspec_if_opentype:TF` 23
`\fontspec_if_script:n` 73
`\fontspec_if_script:nTF` 73
`\fontspec_if_small_caps:` 189
`\fontspec_if_small_caps:TF` 189
`\fontspec_maybe_setup_maths:` 94
`\fontspec_new_lang:nn` 118, 330
`\fontspec_new_script:nn` 114, 283
`\fontspec_parse_colour:niii` . 437, 445
`\fontspec_parse_cv:w` 239, 252
`_fontspec_parse_whitespace:w` 350, 353
`\fontspec_select:nn` 43
`\fontspec_set_em_level:n` 124
`\fontspec_set_family:Nnn`
 3, 29, 42, 55, 103, 146
`\fontspec_set_fontface:NNnn` 159
`\fontspec_set_strong_level:n` ... 171
`\fontspec_setup_maths:` 1, 134
`\fontspec_tmp:` 60, 63
fontspec internal commands:
 `\l_fontspec_check_bool`
 89, 93, 99, 108, 140, 148, 155, 164
 `_fontspec_update_featstr:n` 97
`\FONTSPECCTX` 2
`\FontspecSetCheckBoolFalse` 65
`\FontspecSetCheckBoolTrue` 65
fp commands:
 `\fp_eval:n` 306, 322
 `\fp_new:N` 40, 41

G

`\g` 117
`\Gamma` 52
`\gdef` 2
`\GetFileInfo` 54
`\global` 7
`\grave` 19
group commands:
 `\group_begin:` 4, 28, 37, 81, 131, 151,
 173, 268, 289, 296, 311, 389, 398, 521
 `\group_end:`
 33, 39, 41, 42, 90, 140, 162, 177, 178,
 276, 292, 299, 328, 393, 394, 402, 524

H

`\hat` 25, 115
`\hbar` 71

I

`\IfBooleanTF` 120, 132
`\ifcase` 347
`\ifdefined` 26, 39, 52
`\IfFontExistsTF` 180
`\IfFontFeatureActiveTF` 124, 259
`\IfNoValueTF` 75
`\ifnum` 92, 198, 354
`\ifx` 15, 28, 30, 98, 110, 116, 741, 742
`\ignorespaces` 4, 9, 14, 19, 78, 168
`\InputIfFileExists` 3
int commands:
 `\int_case:nn` 57, 78, 83
 `\int_case:nnTF` 339
 `\int_compare:nNnTF` 146
 `\int_compare:nTF` 32,
 53, 74, 100, 280, 433, 436, 467, 715, 734
 `\int_compare_p:nNn` 90, 141, 196
 `\int_eval:n` 251
 `\int_if_even:nTF` 37
 `\int_incr:N` 88, 96, 109, 138, 152, 153, 203
 `\int_new:N`
 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
 `\int_set:Nn` ... 11, 42, 45, 76, 84, 87,
 94, 94, 105, 124, 137, 143, 149, 149,
 171, 191, 201, 296, 340, 458, 685, 737
 `\int_to_hex:n` 468
 `\int_use:N`
 .. 87, 102, 106, 111, 137, 146, 150, 155
 `\int_zero:N` 70, 88, 114, 129, 139, 158,
 168, 169, 187, 194, 357, 686, 687, 688
 `\l_tmpa_int` 88,
 90, 92, 94, 96, 194, 196, 199, 201, 203
 `\l_tmpb_int` ... 87, 90, 94, 191, 196, 201
`\itdefault`
 . 2, 7, 9, 11, 50, 77, 84, 89, 307, 309,
 532, 533, 537, 566, 567, 572, 587, 589
`\itshape` 47, 75, 121

K

keys commands:
 `\l_keys_choice_int` ... 53, 57, 74, 78, 83
 `\keys_define:mn`
 . 3, 3, 7, 9, 20, 20, 22, 28, 48, 69, 69,
 81, 92, 172, 204, 214, 215, 223, 232,
 239, 246, 246, 259, 269, 278, 285,
 325, 332, 415, 526, 534, 542, 550, 572
 `\keys_if_choice_exist:nmnTF` . 182, 192

264, 265, 266, 267, 268, 269, 270,	
271, 272, 273, 274, 275, 276, 277,	
278, 279, 280, 281, 282, 283, 284,	
285, 286, 287, 288, 289, 290, 291,	
292, 293, 294, 295, 296, 297, 298,	
299, 300, 301, 302, 303, 304, 305,	
306, 307, 308, 309, 310, 311, 312,	
313, 314, 315, 316, 317, 318, 319,	
320, 321, 322, 323, 324, 325, 326,	
327, 328, 329, 330, 331, 332, 333,	
334, 335, 336, 337, 338, 339, 340,	
341, 342, 343, 344, 345, 346, 347,	
348, 349, 350, 351, 352, 353, 354,	
355, 356, 357, 358, 359, 360, 361,	
362, 363, 364, 365, 366, 367, 368,	
369, 370, 371, 372, 373, 374, 375,	
376, 377, 378, 379, 380, 381, 382, 383	
<code>\newfontscript</code> 1, 2, 3, 4, 5, 6, 7,	
8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,	
19, 20, 21, 22, 23, 24, 25, 26, 27, 28,	
29, 30, 31, 32, 33, 34, 35, 36, 37, 38,	
39, 40, 41, 42, 43, 44, 45, 46, 47, 48,	
49, 50, 51, 52, 53, 54, 55, 56, 57, 58,	
59, 60, 61, 62, 63, 64, 65, 66, 67, 68,	
69, 70, 71, 72, 73, 74, 75, 76, 77, 78,	
79, 80, 81, 82, 83, 84, 85, 86, 87, 88,	
89, 90, 91, 92, 93, 94, 95, 96, 97, 98,	
99, 100, 101, 102, 103, 104, 105, 106,	
107, 108, 109, 110, 111, 112, 112,	
113, 114, 115, 116, 117, 118, 119,	
120, 121, 122, 123, 124, 125, 126,	
127, 128, 129, 130, 131, 132, 133,	
134, 135, 136, 137, 138, 139, 140,	
141, 142, 143, 144, 145, 146, 147,	
148, 149, 150, 151, 152, 153, 154, 155	
<code>\newICUfeature</code> 100	
<code>\newopentypefeature</code> 96, 187	
<code>\normalfont</code> 35, 48, 61, 82, 132, 167	
<code>\normalsize</code> 50, 522	
<code>\nullfont</code> 15	
<code>\numexpr</code> 44	
O	
<code>\oldstylenums</code> 2, 35, 128, 287	
<code>\Omega</code> 62	
<code>\or</code> 350, 358, 362	
P	
<code>Path</code> 24	
<code>\Phi</code> 60	
<code>\Pi</code> 57	
<code>\postexhyphenchar</code> 688	
<code>\posthyphenchar</code> 686	
<code>\preexhyphenchar</code> 687	
<code>\prehyphenchar</code> 685	
prg commands:	
<code>\prg_new_conditional:Nnn</code>	
. 1, 5, 17, 23, 26, 34, 44, 58, 60,	
73, 75, 86, 103, 116, 121, 125, 131,	
169, 171, 174, 181, 189, 220, 265, 387	
<code>\prg_return_false:</code> 3, 13, 16, 18, 20,	
24, 28, 28, 29, 30, 31, 41, 49, 50, 51,	
53, 57, 67, 68, 69, 71, 80, 82, 82, 84,	
97, 99, 99, 101, 110, 112, 114, 115,	
123, 125, 127, 129, 132, 140, 142,	
144, 155, 164, 172, 178, 180, 187,	
204, 206, 207, 217, 231, 281, 284, 393	
<code>\prg_return_true:</code> 3, 13,	
16, 24, 28, 28, 29, 42, 47, 50, 54, 67,	
68, 79, 80, 97, 99, 110, 111, 123, 125,	
129, 140, 155, 164, 172, 177, 177,	
187, 205, 206, 217, 237, 243, 284, 394	
<code>\prg_set_conditional:Nnn</code> 13, 21, 26, 35	
<code>\ProcessOptions</code> 22	
prop commands:	
<code>\prop_gclear:N</code> 69, 128	
<code>\prop_get:NnN</code>	
41, 44, 47, 48, 93, 95, 123, 138, 153, 154	
<code>\prop_get:NnNTF</code> 91, 92, 102, 103,	
105, 111, 134, 147, 155, 228, 248, 408	
<code>\prop_gput:Nnn</code> 11, 87,	
90, 137, 137, 142, 222, 223, 224, 225,	
226, 227, 228, 229, 230, 231, 232,	
233, 234, 235, 236, 237, 238, 239,	
240, 241, 242, 243, 244, 245, 246,	
247, 248, 249, 250, 251, 252, 253,	
254, 254, 255, 256, 256, 257, 258,	
259, 260, 261, 262, 263, 264, 265,	
266, 267, 268, 269, 270, 270, 271,	
271, 272, 272, 273, 274, 275, 276,	
277, 277, 278, 278, 279, 279, 280,	
280, 281, 282, 283, 284, 285, 286,	
287, 288, 289, 290, 291, 292, 293,	
294, 295, 296, 297, 298, 299, 300,	
301, 302, 303, 304, 305, 306, 307,	
308, 309, 310, 311, 312, 313, 314,	
315, 316, 317, 318, 319, 320, 321,	
322, 323, 324, 325, 326, 327, 328,	
329, 330, 331, 332, 333, 334, 335,	
336, 337, 338, 339, 340, 341, 342, 343	
<code>\prop_gput_if_new:Nnn</code> 86, 89, 136	
<code>\prop_gremove:Nn</code> 141	

<code>\prop_if_in:NnTF</code>	9, 87	<code>\setmathrm</code>	21, <u>63</u> , 94, 117
<code>\prop_map_inline:Nn</code>	311	<code>\setmathsf</code>	29, <u>75</u> , 96
<code>\prop_new:N</code> 68, 69, 70, <u>71</u> , 72, 73, 74, 75, 76, 269	<code>\setmathtt</code>	33, <u>81</u> , 97
<code>\prop_put:Nnn</code>	87, 88, 171, 204, 334	<code>\setmonofont</code>	16, <u>50</u>
<code>\providecommand</code>	2, 3	<code>\setromanfont</code>	<u>37</u>
<code>\ProvideDocumentCommand</code>	55, 71	<code>\setsansfont</code>	11, <u>37</u>
<code>\providefontface</code>	69	<code>\SetSymbolFont</code>	17, 75, 81
<code>\providefontfamily</code>	53	<code>\settoheight</code>	336
<code>\ProvidesExplFile</code>	48	<code>\sfdefault</code>	43, 45, 46, 99, 123
<code>\ProvidesExplPackage</code>	43, 44, 45	<code>\sffamily</code>	<u>7</u>
<code>\Psi</code>	61	<code>\shapedefault</code> .	13, 14, 74, 75, 78, 79, 80, 81, 87, 88, 91, 92, 203, 305, 306, 585, 586
Q		<code>\Sigma</code>	58
<code>\quad</code>	55	<code>\sishape</code>	<u>35</u>
quark commands:		<code>\sldefault</code>	3, 8, 10, 12, 55, 308, 310, 533, 536, 567, 571, 588, 590
<code>\q_nil</code>	72, 73, 198, 199, 240, 253, 704, 707, 710, 711, 713, 723, 726, 729, 730, 732	<code>\slshape</code>	<u>47</u> , 73
<code>\q_stop</code>	350, 353	<code>\space</code>	34, 39, 44, 192
R		str commands:	
<code>\relax</code>	29, 37, 44, 54, 59, 64, 354	<code>\c_backslash_str</code>	78
<code>\renewcommand</code>	71	<code>\c_colon_str</code>	240, 253
<code>\RenewDocumentCommand</code>	47, 63, 130	<code>\str_case:nn</code>	74, 583, 595
<code>\renewfontface</code>	61	<code>\str_case:nnTF</code>	201, 296
<code>\renewfontfamily</code>	45	<code>\str_case_e:nnTF</code>	375
<code>\RequirePackage</code>	5, 42, 47, 48, 62	<code>\str_if_eq:nnTF</code> 32, 45, 58, 72, 93, 116, 124, 139, 156, 186, 216, 338, 392, 404, 484
<code>\RequirePackageWithOptions</code>	7, 12	<code>\str_if_eq_p:nn</code> .	306, 314, 315, 316, 532, 533, 544, 545, 566, 567, 709, 728
<code>\rmdefault</code>	29, 30, 32, 45, 98, 117, 122	<code>\str_lowercase:n</code>	72, 116
<code>\rmfamily</code>	<u>7</u> , 80, 338	<code>\string</code>	21, 56, 76, 96
S		<code>\strong</code>	<u>163</u>
scan commands:		<code>\strongenv</code>	<u>142</u> , <u>163</u>
<code>\scan_stop:</code>	3, 7, 46, 54	<code>\strongfontdeclare</code>	<u>126</u> , 172
<code>\scdefault</code>	2, 3, 7, 8, 9, 10, 13, 14, 60, 514, 515, 516, 597, 598	<code>\strongreset</code>	133, 159, <u>163</u>
<code>\scitdefault</code> .	2, 7, 9, 11, 12, 13, 38, 599, 600	<code>\suppressfontnotfounderror</code>	11
<code>\scshape</code>	<u>47</u>	sys commands:	
<code>\scsldefault</code> .	3, 8, 10, 11, 12, 14, 601, 602	<code>\sys_if_engine luatex:TF</code>	3
<code>\select</code>	19	<code>\sys_if_engine xetex:TF</code>	10
<code>\selectfont</code>	5, 38, 44, 45, 110, 163, 315	T	
seq commands:		T _E X and L ^A T _E X 2 _ε commands:	
<code>\seq_if_empty:NTF</code>	160	<code>\@</code>	31, 61
<code>\seq_new:N</code>	45	<code>\@filelist</code>	50
<code>\seq_put_right:Nn</code>	148, 163	<code>\@ifpackageloaded</code> 1, 6, 13, 14, 15, 96, 100, 101, 102, 103, 104, 105, 106, 107, 108, 112, 113, 114, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130
<code>\setboldmathrm</code>	25, 28, <u>69</u> , 95, 117	<code>\@nomath</code>	94, 144
<code>\setfontface</code>	65		
<code>\setfontfamily</code>	49		
<code>\setmainfont</code>	6, 24, <u>24</u> , 28, 115		
<code>\SetMathAlphabet</code> 77, 78, 79, 80, 84, 87, 88, 89, 91, 92		

<code>\@onlypreamble</code>	94, 95, 96, 97	<code>\tl_if_empty:NTF</code>	
<code>\@rmfamilyhook</code>	7, 9	29, 46, 50, 82, 91, 181,
<code>\@sffamilyhook</code>	10		213, 226, 238, 281, 305, 332, 353,
<code>\@tempa</code>	29, 30		361, 365, 378, 446, 459, 487, 506,
<code>\@ttfamilyhook</code>	11		555, 561, 583, 618, 619, 626, 627, 628
<code>\add@unicode@accent</code>	11	<code>\tl_if_empty:nTF</code> 7, 14, 18, 58, 95, 96,	
<code>\color@</code>	428	97, 132, 140, 152, 187, 328, 355, 376, 606
<code>\curr@fontshape</code>	104, 161, 214	<code>\tl_if_empty_p:n</code>	81, 94, 131, 179
<code>\define@antt@mathversions</code>	98	<code>\tl_if_eq:NNTF</code>	201, 448, 460
<code>\define@iwona@mathversions</code>	110	<code>\tl_if_eq:nTF</code>	98, 166
<code>\define@kurier@mathversions</code>	116	<code>\tl_if_exist:NTF</code>	146, 514
<code>\f@encoding</code>	24, 199, 202, 203	<code>\tl_if_exist_p:N</code>	20
<code>\f@family</code>	3, 3, 24, 41, 44, 47, 48,	<code>\tl_if_in:NnTF</code>	50, 50, 283
	93, 95, 123, 138, 153, 154, 199, 202, 203	<code>\tl_if_in:nTF</code>	65, 73, 75, 185
<code>\f@series</code> 24, 136, 147, 150, 199, 202, 203		<code>\tl_if_single:nTF</code>	128, 412
<code>\f@shape</code>	20, 25, 44, 86, 95, 193	<code>\tl_new:N</code>	77, 78, 79, 83, 84, 85, 86,
<code>\f@size</code>	39, 104,	87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
	139, 146, 161, 214, 390, 391, 412, 523		97, 98, 99, 100, 101, 102, 103, 104,
<code>\not@math@alphabet</code>	37, 49, 54, 59, 64		105, 106, 107, 108, 109, 110, 111,
<code>\reset@font</code>	165		112, 113, 114, 115, 116, 117, 118,
<code>\two@digits</code>	231, 243, 244		119, 120, 121, 125, 126, 127, 128,
tex commands:			129, 130, 131, 132, 133, 134, 135,
<code>\tex_font:D</code>	59		136, 138, 139, 141, 142, 143, 144,
<code>\tex_hyphenchar:D</code>	54		145, 146, 146, 147, 148, 149, 153,
<code>\tex_iffontchar:D</code>	46		154, 155, 156, 157, 158, 159, 160
<code>\textsc</code>	36	<code>\tl_put_left:Nn</code>	46
<code>\textsf</code>	32	<code>\tl_put_right:Nn</code>	
<code>\textsi</code>	35	9, 10, 11, 136, 396, 406, 418
<code>\Theta</code>	54	<code>\tl_remove_all:Nn</code>	47, 86, 235, 322
<code>\tilde</code>	21	<code>\tl_remove_once:Nn</code>	52
<code>\title</code>	31	<code>\tl_replace_all:Nnn</code>	
tl commands:		14, 16, 18, 99, 99, 321
<code>\c_empty_tl</code>		<code>\tl_set:Nn</code>	21, 21, 22, 22, 23, 26, 28,
.	61, 710, 711, 729, 730, 741, 742	29, 34, 39, 39, 42, 45, 46, 46, 47, 49,
<code>\tl_build_begin:N</code>	423, 424		53, 53, 55, 76, 81, 85, 95, 99, 100, 104,
<code>\tl_build_end:N</code>	429, 430		105, 107, 112, 118, 119, 123, 124,
<code>\tl_build_put_right:Nn</code>	492		129, 137, 140, 149, 150, 154, 160,
<code>\tl_clear:N</code>			161, 162, 163, 164, 167, 196, 214,
.	23, 24, 45, 107, 135, 277, 287, 308,		215, 234, 239, 243, 250, 253, 255,
	441, 676, 677, 678, 679, 694, 698, 699		273, 294, 295, 301, 305, 306, 320,
<code>\tl_clear_new:N</code>	80, 81, 82		320, 326, 329, 339, 353, 356, 357,
<code>\tl_const:Nn</code>	7,		361, 365, 365, 379, 381, 384, 389,
	8, 9, 10, 11, 12, 13, 14, 150, 151, 152		413, 414, 434, 447, 453, 465, 474,
<code>\tl_count:n</code>	433, 436, 715, 734		478, 489, 506, 523, 556, 578, 613, 684
<code>\tl_gclear:N</code>	267, 680, 681, 693	<code>\tl_set_eq:NN</code>	30, 31, 33, 41, 43, 44,
<code>\tl_gput_right:Nn</code>	580, 657	46, 48, 49, 51, 52, 53, 55, 56, 56, 57,
<code>\tl_gremove_all:Nn</code>	666		59, 126, 157, 168, 170, 193, 286, 425,
<code>\tl_gset:Nn</code>	41, 42,		442, 558, 566, 580, 588, 695, 696, 697
	44, 60, 88, 98, 99, 100, 122, 123, 124,	<code>\tl_to_str:N</code>	21
	147, 162, 255, 269, 298, 341, 358, 655	<code>\tl_to_str:n</code>	78, 186
<code>\tl_gset_eq:NN</code>	156, 169, 230, 241, 682	<code>\tl_trim_spaces:n</code>	14, 16

