ΠΟΛΥΓΛΩΣΣΙΑ

Polyglossia: Modern multilingual typesetting with XƎLaTeX and LuaLaTeX

François Charette Arthur Reutenauer[*]
Bastien Roucariès Jürgen Spitzmüller

2020/04/08 v1.49
(pdf file generated on 9 October 2020)

Contents

[*]Current maintainer

1 Introduction

Polyglossia is a package for facilitating multilingual typesetting with XƎLaTeX and LuaLaTeX. Basically, it can be used as an alternative to babel for performing the following tasks automatically:

1. Loading the appropriate hyphenation patterns.
2. Setting the script and language tags of the current font (if possible and available), via the package fontspec.
3. Switching to a font assigned by the user to a particular script or language.
4. Adjusting some typographical conventions according to the current language (such as afterindent, frenchindent, spaces before or after punctuation marks, etc.).
5. Redefining all document strings (like "chapter", "figure", "bibliography").
6. Adapting the formatting of dates (for non-Gregorian calendars via external packages bundled with polyglossia: currently the Hebrew, Islamic and Farsi calendars are supported).
7. For languages that have their own numbering system, modifying the formatting of numbers appropriately (this also includes redefining the alphabetic sequence for non-Latin alphabets).[1]
8. Ensuring proper directionality if the document contains languages that are written from right to left (via the package bidi, available separately).

Several features of babel that do not make sense in the XƎTeX world (like font encodings, shorthands, etc.) are not supported. Generally speaking, polyglossia aims to remain as compatible as possible with the fundamental features of babel while being cleaner, light-weight, and modern. The package antomega has been very beneficial in our attempt to reach this objective.

Requirements The current version of polyglossia makes use of some convenient macros defined in the etoolbox package by Philipp Lehmann and Joseph Wright. Being designed for XƎLaTeX and LuaLaTeX, it obviously also relies on fontspec by Will Robertson. For languages written from right to left, it needs the package bidi (for XƎTeX) or luabidi (for LuaTeX) by Vafa Khalighi (وفا خليقي) and the bidi-tex GitHub Organisation. Polyglossia also bundles three packages for calendaric computations (hebrewcal, hijrical, and farsical).

[1]This is done by bundled sub-packages such as arabicnumbers.

2 Setting up multilingual documents

2.1 Activating languages

The default language of a document is specified by means of the command

`\setdefaultlanguage`

```
\setdefaultlanguage[⟨options⟩]{⟨lang⟩}
```

`\setmainlanguage`

(or, equivalently, `\setmainlanguage`). Secondary languages are specified with

`\setotherlanguage`

```
\setotherlanguage[⟨options⟩]{⟨lang⟩}.
```

All these commands allow you to set language-specific options.[2] It is also possible to load a series of secondary languages at once (but without options) using

`\setotherlanguages`

```
\setotherlanguages{⟨lang1⟩,⟨lang2⟩,⟨lang3⟩,⟨…⟩}.
```

All language-specific options can be modified locally by means of the language-switching commands described in section 3.

Note In general, it is advisable to activate the languages *after* all packages have been loaded. This is particularly important if you use right-to-left scripts or languages with babel shorthands.

2.2 Supported languages

Table 1 lists all languages currently supported. Those in red have specific options and/or commands that are explained in section 6 below.

v1.0.1 v1.1.1 v1.2.0 v1.43 v1.45 v1.46

Version Notes The support for Amharic ← should be considered an experimental attempt to port the package ethiop; feedback is welcome. Version 1.1.1 ← added support for Asturian, Lithuanian, and Urdu. Version 1.2 ← introduced Armenian, Occitan, Bengali, Lao, Malayalam, Marathi, Tamil, Telugu, and Turkmen.[3] Version 1.43 ← brought basic support for Japanese (this is considered experimental, feedback is appreciated). In version 1.45 ←, support for Kurdish and Mongolian as well as some new variants (Canadian French and English) have been added. Furthermore, for consistency reasons, some language have been renamed (*farsi*⟶*persian*, *friulan*⟶*friulian*, *magyar*⟶*hungarian*, *portuges*⟶*portuguese*, *samin*⟶*sami*) or merged (*bahasai*/*bahasam*⟶*malay*, *brazil*/*portuges*⟶*portuguese*, *lsorbian*/*usorbian*⟶*sorbian*, *irish*/*scottish*⟶*gaelic*, *norsk*/*nynorsk*⟶*norwegian*). The old names are still supported for backwards compatibility reasons. Version 1.46 ← introduces support for Afrikaans, Belarusian, Bosnian and Georgian.

[2] Section 6 documents these options for the respective languages.

[3] See acknowledgements at the end for due credit to the various contributors.

Table 1. Languages currently supported in polyglossia

afrikaans	danish	hungarian	marathi	slovenian
albanian	divehi	icelandic	mongolian	sorbian
amharic	dutch	interlingua	nko	spanish
arabic	english	italian	norwegian	swedish
armenian	esperanto	japanese	occitan	syriac
asturian	estonian	kannada	persian	tamil
basque	finnish	khmer	piedmontese	telugu
belarusian	french	korean	polish	thai
bengali	friulian	kurdish	portuguese	tibetan
bosnian	gaelic	lao	romanian	turkish
breton	galician	latin	romansh	turkmen
bulgarian	georgian	latvian	russian	ukrainian
catalan	german	lithuanian	sami	urdu
coptic	greek	macedonian	sanskrit	vietnamese
croatian	hebrew	malay	serbian	welsh
czech	hindi	malayalam	slovak	

2.3 Relation to and use of Babel language names

If you are familiar with the babel package, you will note that polyglossia's language naming slightly differs. Whereas babel has a unique name for each language variety (*e.g., american* and *british*), polyglossia differentiates language varieties via language options (*e.g., english*, `variant=american`).

Furthermore, babel sometimes uses abbreviated language names (*e.g., bahasam* for Bahasa Malayu) as well as endonyms, *i.e.,* language names coming from the designated languages (such as *bahasa*, *canadien* or *magyar*). As opposed to this, polyglossia always uses spelled-out (lower-cased) English language names. Please refer to table 2 for the differing language names in both packages.

v1.46 For convenience reasons, polyglossia also supports the use of babel names ← (for the few justified exceptions, please refer to the notes in table 2). The babel names listed in table 2 can be used instead of the corresponding polyglossia name/options in `\setdefaultlanguage` and `\setotherlanguage` as well as in the polyglossia and babel language switching commands/environments documented in section 3.1 and 3.2 (*e.g.,* `\textaustrian` is synonymous to `\textgerman[variant=austrian,spelling=old]`). However, unless you have special reasons, we strongly encourage you to use the polyglossia names.

Table 2. Babel-polyglossia language name matching

Babel name	**Polyglossia name**	**Polyglossia options**
acadien	french	variant=acadian
american	english	variant=american [*default*]
australian	english	variant=australian
austrian	german	variant=austrian, spelling=old
bahasa	malay	variant=indonesian [*default*]
bahasam	malay	variant=malaysian
brazil	portuguese	variant=brazilian
british	english	variant=british
canadian	english	variant=canadian
canadien	french	variant=canadian
classiclatin[a]	latin	variant=classic
farsi	persian	
ecclesiasticlatin[b]	latin	variant=ecclesiastic
friulan	friulian	
german[c]	german	spelling=old
irish	gaelic	variant=irish [*default*]
kurmanji	kurdish	variant=kurmanji
lowersorbian	sorbian	variant=lower
magyar	hungarian	
medievallatin[d]	latin	variant=medieval
naustrian	german	variant=austrian
newzealand	english	variant=newzealand
ngerman	german	variant=german [*default*]
norsk	norwegian	variant=bokmal
nswissgerman	german	variant=swiss
nynorsk	norwegian	variant=nynorsk [*default*]
polutonikogreek	greek	variant=polytonic
portuges	portuguese	variant=portuguese [*default*]
samin	sami	
scottish	gaelic	variant=scottish
serbianc	serbian	script=Cyrillic
slovene	slovenian	
spanishmx	spanish	variant=mexican
swissgerman	german	variant=swiss, spelling=old
uppersorbian	sorbian	variant=upper [*default*]

[a]In babel currently only selectable via dot modifier (*latin.classic*).
[b]In babel currently only selectable via dot modifier (*latin.ecclesiastic*).
[c]Due to the name conflict only available in polyglossia as *germanb* (which is a babel synonym).
[d]In babel currently only selectable via dot modifier (*latin.medieval*).

2.4 Using IETF language tags

v1.47 Polyglossia ← also supports the use of language tags that conform to the IETF BCP-47 *Best Current Practice*.[4] Thus, you can use tags such as `en-GB` (for British English) or `de-AT-1901` (for Austrian German, old spelling) in `\setdefaultlanguage` and `\setotherlanguage` as well as in the language switching command `\textlang{⟨tag⟩}`, the environment `\begin{lang}{⟨tag⟩}` … `\end{lang}` and the babel language switching commands/environments documented in section 3.2. Table 3 lists the currently supported tags.

Table 3. BCP47-polyglossia language name matching

BCP-47 tag	Polyglossia name	Polyglossia options
aeb	arabic	locale=tunisia
af	afrikaans	
afb	arabic	locale=default
am	amharic	
apd	arabic	locale=default
ar	arabic	
ar-IQ	arabic	locale=mashriq
ar-JO	arabic	locale=mashriq
ar-LB	arabic	locale=mashriq
ar-MR	arabic	locale=mauritania
ar-PS	arabic	locale=mashriq
ar-SY	arabic	locale=mashriq
ar-YE	arabic	locale=default
arq	arabic	locale=algeria
ary	arabic	locale=morocco
arz	arabic	locale=default
ast	asturian	
ayl	arabic	locale=libya
be	belarusian	
be-tarask	belarusian	spelling=classic
bg	bulgarian	
bn	bengali	
bo	tibetan	
br	breton	
bs	bosnian	
ca	catalan	

[4] Please refer to `https://tools.ietf.org/html/bcp47` and `https://en.wikipedia.org/wiki/IETF_language_tag` for details.

Table 3. BCP47-polyglossia language name matching (*continued*)

BCP-47 tag	Polyglossia name	Polyglossia options
ckb	kurdish	variant=sorani [*default*]
ckb-Arab	kurdish	variant=sorani, script=Arabic [*default*]
ckb-Latn	kurdish	variant=sorani, script=Latin
cop	coptic	
cy	welsh	
cz	czech	
da	danish	
de	german	
de-AT	german	variant=austrian, spelling=new
de-AT-1901	german	variant=austrian, spelling=old
de-AT-1996	german	variant=austrian, spelling=new
de-CH	german	variant=swiss, spelling=new
de-CH-1901	german	variant=swiss, spelling=old
de-CH-1996	german	variant=swiss, spelling=new
de-DE	german	variant=german, spelling=new
de-DE-1901	german	variant=german, spelling=old
de-DE-1996	german	variant=german, spelling=new [*default*]
de-Latf	german	script=blackletter
de-Latf-AT	german	variant=austrian, spelling=new, script=blackletter
de-Latf-AT-1901	german	variant=austrian, spelling=old, script=blackletter
de-Latf-AT-1996	german	variant=austrian, spelling=new, script=blackletter
de-Latf-CH	german	variant=swiss, spelling=new, script=blackletter
de-Latf-CH-1901	german	variant=swiss, spelling=old, script=blackletter
de-Latf-CH-1996	german	variant=swiss, spelling=new, script=blackletter
de-Latf-DE	german	variant=german, spelling=new, script=blackletter
de-Latf-DE-1901	german	variant=german, spelling=old, script=blackletter
de-Latf-DE-1996	german	variant=german, spelling=new, script=blackletter
dsb	sorbian	variant=lower
dv	divehi	
el	greek	
el-monoton	greek	variant=monotonic [*default*]
el-polyton	greek	varant=polytonic
en	english	
en-AU	english	variant=australian
en-CA	english	variant=canadian
en-GB	english	variant=british
en-NZ	english	variant=newzealand
en-US	english	variant=us [*default*]
eo	esperanto	

Table 3. BCP47-polyglossia language name matching (*continued*)

BCP-47 tag	Polyglossia name	Polyglossia options
es	spanish	
es-ES	spanish	variant=spanish [*default*]
es-MX	spanish	variant=mexican
et	estonian	
eu	basque	
fa	persian	
fi	finnish	
fr	french	
fr-CA	french	variant=canadian
fr-CH	french	variant=swiss
fr-FR	french	variant=french [*default*]
fur	friulian	
ga	gaelic	variant=irish [*default*]
gd	gaelic	variant=scottish
gl	galician	
grc	greek	variant=ancient
he	hebrew	
hi	hindi	
hr	croatian	
hsb	sorbian	variant=upper [*default*]
hu	hungarian	
hy	armenian	
ia	interlingua	
id	malay	variant=indonesian
is	icelandic	
it	italian	
ja	japanese	
ka	georgian	
km	khmer	
kmr	kurdish	variant=kurmanji
kmr-Arab	kurdish	variant=kurmanji, script=Arabic
kmr-Latn	kurdish	variant=kurmanji, script=Latin
kn	kannada	
ko	korean	
ku	kurdish	
ku-Arab	kurdish	script=Arabic
ku-Latn	kurdish	script=Latin
la	latin	
la-x-classic	latin	variant=classic

Table 3. BCP47-polyglossia language name matching (*continued*)

BCP-47 tag	Polyglossia name	Polyglossia options
la-x-ecclesia	latin	variant=ecclesiastic
la-x-medieval	latin	variant=medieval
lo	lao	
lt	lithuanian	
lv	latvian	
mk	macedonian	
ml	malayalam	
mn	mongolian	
mr	marathi	
nb	norwegian	variant=bokmal
nko	nko	
nl	dutch	
nn	norwegian	variant=nynorsk [*default*]
oc	occitan	
pl	polish	
pms	piedmontese	
pt	portuguese	
pt-BR	portuguese	variant=brazilian
pt-PT	portuguese	variant=portuguese [*default*]
rm	romansh	
ro	romanian	
ru	russian	
ru-luna1918	russian	spelling=modern [*default*]
ru-petr1708	russian	spelling=old
sa	sanskrit	
sa-Beng	sanskrit	script=Bengali
sa-Deva	sanskrit	script=Devanagari [*default*]
sa-Gujr	sanskrit	script=Gujarati
sa-Knda	sanskrit	script=Kannada
sa-Mlym	sanskrit	script=Malayalam
sa-Telu	sanskrit	script=Telugu
se	sami	
sk	slovak	
sl	slovenian	
sq	albanian	
sr	serbian	
sr-Cyrl	serbian	script=Cyrillic
sr-Latn	serbian	script=Latin [*default*]
sv	swedish	

Table 3. BCP47-polyglossia language name matching (*continued*)

BCP-47 tag	Polyglossia name	Polyglossia options
syr	syriac	
ta	tamil	
te	telugu	
th	thai	
tk	turkmen	
tr	turkish	
uk	ukrainian	
ur	urdu	
vi	vietnamese	
zsm	malay	variant=malaysian [*default*]

2.5 Global options

Polyglossia can be loaded with the following global package options:

- **`babelshorthands`** ⟵ `= *true` or *`false`* v1.1.1
  Globally activates babel shorthands whenever available. Currently shorthands are implemented for Afrikaans, Belarusian, Catalan, Croatian, Czech, Dutch, Finnish, Georgian, German, Italian, Latin, Mongolian, Russian, Slovak, and Ukrainian. Please refer to the respective language descriptions (sec. 6) for details.
- **`localmarks`** `= *true` or *`false`*
  redefines the internal LaTeX macros `\markboth` and `\markright` to the effect that the header text is explicitly set in the currently active language (*i.e.,* wrapped into `\foreignlanguage{⟨lang⟩}{⟨…⟩}`).
  v1.2.0 In earlier versions of polyglossia, ⟵ this option was enabled by default, but we now realize that it causes more problems than it helps (since it breaks if a package or class redefines `\markboth` or `\markright`), so it is now disabled by default. For backwards compatibility, the option **`nolocalmarks`** which used to switch off the previous default, and now equals the default, is still available.
- **`luatexrenderer`** ⟵ `= ⟨renderer⟩` (default value: `Harfbuzz`) v1.50
  determines which font renderer is used with LuaTeX output. The correct font renderer is essential particularly for non-Latin scripts. By default, polyglossia uses the `Harfbuzz` renderer that has been introduced to LuaTeX in 2019 (TeXLive 2020), as this gives the best results generally. If you want

to use a different renderer, you can specify this here (or individually for specific fonts via the optional argument of the font selection commands). Please refer to the fontspec manual for supported values and for details on how to change the renderer for individual fonts.
luatexrenderer=none disables polyglossia's automatic renderer setting.

- **verbose** = **true* or false
determines whether info messages and (some of the) warnings issued by LaTeX, fontspec and polyglossia are output.

3 Language-switching commands

3.1 Recommended commands

\text⟨lang⟩ \textlang v1.46 For each activated language the command `\text⟨lang⟩[⟨options⟩]{⟨…⟩}` (as well as the synonymous `\textlang[⟨options⟩]{⟨lang⟩}{⟨…⟩}` ⟵) becomes available for short insertions of text in that language.

For example `\textrussian{\today}` and `\textlang{russian}{\today}` yield 9 октября 2020 г. The commands switch to the correct hyphenation patterns, they activate some extra features for the selected language (such as extra spacing before punctuation in French), and they translate the date when using `\today`. They do not, however, translate so-called *caption strings*, *i.e.*, "chapter", "figure" etc., to the local language (these remain in the currently active 'outer' language).

⟨lang⟩ lang v1.47 The environment ⟨lang⟩, which is also available for any activated language (as well as the equivalent `\begin{lang}[⟨options⟩]{⟨lang⟩}` … `\end{lang}` ⟵), is meant for longer passages of text. It behaves slightly different than the `\text⟨lang⟩` and `\textlang` commands: It does everything the latter do, but additionally, the caption strings are translated as well, and the language is also passed to auxiliary files, the table of contents and the lists of figures/tables. Like the commands, the environment provides the possibility of setting language options locally. For instance the following allows us to quote the beginning of Homer's *Iliad*:

```
\begin{quote}
 \begin{greek}[variant=ancient]
   μῆνιν ἄειδε θεὰ Πηληϊάδεω Ἀχιλῆος οὐλομένην, ἣ μυρί' Ἀχαιοῖς
   ἄλγε' ἔθηκε, πολλὰς δ' ἰφθίμους ψυχὰς Ἄϊδι προΐαψεν ἡρώων,
   αὐτοὺς δὲ ἑλώρια τεῦχε κύνεσσιν οἰωνοῖσί τε πᾶσι, Διὸς δ'
   ἐτελείετο βουλή, ἐξ οὗ δὴ τὰ πρῶτα διαστήτην ἐρίσαντε Ἀτρείδης
```

```
    τε ἄναξ ἀνδρῶν καὶ δῖος Ἀχιλλεύς.
  \end{greek}
\end{quote}
```

> μῆνιν ἄειδε θεὰ Πηληϊάδεω Ἀχιλῆος οὐλομένην, ἣ μυρί' Ἀχαιοῖς ἄλγε' ἔθηκε, πολλὰς δ' ἰφθίμους ψυχὰς Ἄϊδι προΐαψεν ἡρώων, αὐτοὺς δὲ ἑλώρια τεῦχε κύνεσσιν οἰωνοῖσί τε πᾶσι, Διὸς δ' ἐτελείετο βουλή, ἐξ οὗ δὴ τὰ πρῶτα διαστήτην ἐρίσαντε Ἀτρεΐδης τε ἄναξ ἀνδρῶν καὶ δῖος Ἀχιλλεύς.

`Arabic` Note that for Arabic one cannot use the environment `arabic`, as `\arabic` is defined internally by LaTeX. In this case we need to use the environment `Arabic` instead.

3.2 Babel commands

Some macros defined in babel's `hyphen.cfg` (and thus usually compiled into the XeLaTeX and LuaLaTeX format) are redefined, but keep a similar behaviour.

`\selectlanguage`
`\foreignlanguage`
`otherlanguage`
`otherlanguage*`
`hyphenrules`
v1.50

- `\selectlanguage[⟨options⟩]{⟨lang⟩}`
- `\foreignlanguage[⟨options⟩]{⟨lang⟩}{⟨…⟩}`
- `\begin{otherlanguage}[⟨options⟩]{⟨lang⟩}` … `\end{otherlanguage}`
- `\begin{otherlanguage*}[⟨options⟩]{⟨lang⟩}` … `\end{otherlanguage*}`
- `\begin{hyphenrules}[⟨options⟩]{⟨lang⟩}` … `\end{hyphenrules}` ←

`\selectlanguage{⟨lang⟩}` and the `otherlanguage` environment are identical to the `⟨lang⟩` environment, except that `\selectlanguage{⟨lang⟩}` does not need to be explicitly closed. The command `\foreinlanguage{⟨lang⟩}{⟨…⟩}` and the `otherlanguage*` environment are identical with the use of the `\text⟨lang⟩` or `\textlang` command, with the one notable exception that they do not translate the date with `\today`.

The `⟨hyphenrules⟩` environment only switches the hyphenation patterns to the one associated with the language `⟨lang⟩` (or the language variety as specified via `⟨options⟩`). It does no further language-specific change.

Since the XeLaTeX and LuaLaTeX format incorporate babel's `hyphen.cfg`, the low-level commands for hyphenation and language switching defined there are in principal also accessible. Note, however, that the availability of such low-level commands is not guaranteed, as `hyphen.cfg`, which is out of polyglossia's control, is (or at least has been) subject to regular change.

3.3 Other commands

The following commands are probably of lesser interest to the end user, but ought to be mentioned here.

`\selectbackgroundlanguage`

- `\selectbackgroundlanguage{⟨lang⟩}`: this selects the global font setup and the numbering definitions for the default language.

`\resetdefaultlanguage`

- `\resetdefaultlanguage[⟨options⟩]{⟨lang⟩}` (experimental): completely switches the default language to another one in the middle of a document: *this may have adverse effects*!

`\normalfontlatin` `\rmfamilylatin` `\sffamilylatin` `\ttfamilylatin`

- `\normalfontlatin`: in an environment where `\normalfont` has been redefined to a non-latin script, this will reset to the font defined with `\setmainfont` etc. In a similar vein, it is possible to use `\rmfamilylatin`, `\sffamilylatin`, and `\ttfamilylatin`.

`\latinalph`

- `\latinalph`: Representation of counter as a lower-case letter: 1 = a, 2 = b, etc.

`\latinAlph`

- `\latinAlph`: Representation of counter as a upper-case letter: 1 = A, 2 = B, etc.

3.4 Setting up alias commands

By means of the macro

`\setlanguagealias` v1.46

```
\setlanguagealias[⟨options⟩]{⟨language⟩}{⟨alias⟩}
```
←

you can define alias commands for specific language (variants). *E.g.,*

```
\setlanguagealias[variant=austrian]{german}{AT}
```

will define a command `\textAT` as well as an environment `{AT}` which will link towards the command `\textgerman[variant=austrian]` and the environment `{german}[variant=austrian]`, respectively. The aliases can also be used in the language switching commands described in section 3.1 and 3.2. Note, though, that the usual restrictions for command names apply, so something such as `de-AT` or `de_AT` will not work since `-` and `_` are not allowed in command names (the same holds true for any non-ASCII character and for digits).

For the latter case, and for the case where an alias would clash with an existing command (*e.g.,* `\fi`) or a `\text⟨…⟩` command (*e.g.,* `\textit`), a starred version `\setlanguagealias*` is provided which does neither define a `\text⟨alias⟩` command nor an `⟨alias⟩` environment, but which will set up the alias for everything else, including `\textlang{⟨alias⟩}` and `\begin{lang}{⟨alias⟩}`.

`\setlanguagealias*`

Polyglossia comes with some aliases predefined, namely aliases for babel language names (see sec. 2.3) and for IETF BCP-47 language tags (the latter via `\setlanguagealias*`; see sec. 2.4).

4 Font setup

With polyglossia it is possible to associate a specific font with any script or language that occurs in the document. That font should always be defined as `\⟨script⟩font` or `\⟨language⟩font`. For instance, if the default font defined by `\setmainfont` does not support Greek, then one can define the font used to display Greek with:

```
\newfontfamily\greekfont[Script=Greek,⟨…⟩]{⟨font⟩}.
```

Note that polyglossia will use the font defined as is, so assure to do all necessary settings (please refer to the fontspec documentation for details). For instance, if `\arabicfont` is explicitly defined, then the option `Script=Arabic` should be included in that definition.

v1.2.0 If a specific sans serif or monospace ('teletype') font is needed for a particular script or language, it can be defined by means of ← `\⟨script⟩fontsf` or `\⟨language⟩fontsf` and `\⟨script⟩fonttt` or `\⟨language⟩fonttt`, respectively.

Whenever a new language is activated, polyglossia will first check whether a font has been defined for that language or – for languages in non-Latin scripts – for the script it uses. If it is not defined, it will use the currently active font and – in the case of OpenType fonts – will attempt to turn on the appropriate OpenType tags for the script and language used, in case these are available in the font, by means of fontspec's `\addfontfeature`. If the current font does not appear to support the script of that language, an error message is displayed.

5 Adapting hyphenation

5.1 Hyphenation exceptions

TeX provides the command `\hyphenation{⟨exceptions⟩}` to globally define hyphenation exceptions which override the hyphenation patterns for specified words. The command takes as argument a space-separated list of words where hyphenation points are marked by a dash (if no dash is used, the respective word is not hyphenated at all):

```
\hyphenation{%
  po-ly-glos-sia
  LaTeX
}
```

These exceptions, however, apply to all languages. In addition to this, polyglossia provides the command ←

v1.45 `\pghyphenation`

```
\pghyphenation[⟨options⟩]{⟨lang⟩}{⟨exceptions⟩}
```

which can be used to define exceptions that only apply to a specific language or language variant, respectively.

5.2 Hyphenation thresholds

Polyglossia sets reasonable defaults for the hyphenation thresholds of each language, *i.e.,* the number of characters that must at least be there at the beginning or end of a word before it is hyphenated (`\lefthyphenmin` and `\righthyphenmin` in TeX). For instance, with English, this threshold is 2 at the beginning ('left') and 3 at the end ('right'), so a word will not be hyphenated within the first two characters at the beginning and the last three characters at the end.

To change this value, polyglossia provides the command ←

v1.50 `\setlanghyphenmins`

```
\setlanghyphenmins[⟨options⟩]{⟨lang⟩}{⟨l⟩}{⟨r⟩}
```

where ⟨`lang`⟩ is to be replaced with the respective language name or alias, ⟨`options`⟩ can be used to delimit the modification to a particular variety (*e.g.,* via `variant` or `spelling`), ⟨`l`⟩ with the left threshold value (*e.g.,* `3`), and ⟨`r`⟩ with the right threshold value (*e.g.,* `\setlanghyphenmins[spelling=old]{german}{4}{4}`). This setting can be changed repeatedly in the preamble and the document body. It applies to all subsequent text in the respective language (variety), but only after the next language switch.

5.3 Hyphenation disabling

In some very specific contexts (such as music score creation), TeX hyphenation is something to avoid completely as it may cause troubles. Polyglossia provides two functions: `\disablehyphenation` and `\enablehyphenation`. Note that if you select a new language while hyphenation is disabled, it will remain disabled. If you re-enable it, the hyphenation patterns of the currently selected language will be activated.

`\disablehyphenation`
`\enablehyphenation`

6 Language-specific options and commands

This section gives a list of all languages for which options and end-user commands are defined. Note the following conventions:

- The preset value of each option (*i.e.,* the setting that applies by default, if the option is not explicitly set) is given in *`italics`*.
- If an option key may be used without a value, the value that applies for value-less keys is marked by a preceding `*asterisk`.

For instance, `babelshorthands = *true` or *`false`* means that `babelshorthands` is `false` by default in the respective language, and that passing `babelshorthands` (without value) is equivalent to passing `babelshorthands=true`.

6.1 afrikaans

Options:

v1.1.1

- **`babelshorthands`** ⟵ = `*true` or *`false`*
  If this is turned on, the following shorthands defined for fine-tuning hyphenation and micro-typography of Afrikaans words are activated:

`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-` in default TeX).
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.
`"|`	disables a ligature at this position.
`""`	allows for a line break at this position (without hyphenation sign).
`"/`	a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

6.2 arabic

Options:

- **`calendar`** = *`gregorian`* or `islamic` (= `hijri`)
- **`locale`** = *`default`*[5], `mashriq`[6], `libya`, `algeria`, `tunisia`, `morocco`, `mauritania`

[5]For Egypt, Sudan, Yemen and the Gulf states.
[6]For Iraq, Syria, Jordan, Lebanon and Palestine.

This setting influences the spelling of the month names for the Gregorian calendar, as well as the form of the numerals (unless overriden by the following option).

- **`numerals`** = *`mashriq`* or `maghrib`
  The latter is the default when `locale=algeria`, `tunisia`, or `morocco`.
- v1.50 **`abjadalph`** ← = `*true` or *`false`*
  Set this to true if you want the alphabetic counters to be output using `\abjadalph` rather than `\abjad`. Note that this limits the counter scope to 28 (see `\abjadalph` below).
- v1.0.3 **`abjadjimnotail`** ← = `*true` or *`false`*
  Set this to true if you want the *abjad* form of the number three to be جـ – as in the manuscript tradition – instead of the modern usage ج.

Commands:

- `\abjad` `\abjad` outputs Arabic *abjad* numbers according to the Mashriq varieties. Example: `\abjad{1863}` yields غضسج.
- `\abjadmaghribi` `\abjadmaghribi` outputs Arabic *abjad* numbers according to the Maghrib varieties. Example: `\abjadmaghribi{1863}` yields شظطصج.
- `\abjadalph` v1.50 `\abjadalph` ← steps through the Arabic alphabet, thus it can only be used up to 28. Example: `\textarabic{\abjadalph{18}}` yields ص.
- `\aemph` v1.2.0 `\aemph` to emphasize text with `\overline`. ← `\textarabic{\aemph{اب}}` yields اب. This command is also available for Farsi, Urdu, etc.

6.3 armenian

Options:

- v1.45 **`variant`** ← = `eastern` or *`western`*
- v1.45 **`numerals`** ← = `armenian` or *`arabic`*

6.4 belarusian ← v1.46

Options:

- **`babelshorthands`** = `*true` or *`false`*
  If this is turned on, the following shorthands are activated:

`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).
`"=`	adds an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to

plain `-`).

`"~` for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

`"|` disables a ligature at this position.

`""` allows for a line break at this position (without hyphenation sign).

`",` thinspace for initials with a breakpoint in following surname.

`"‘` for German left double quotes (looks like „).

`"’` for German right double quotes (looks like “).

`"<` for French left double quotes (looks like «).

`">` for French right double quotes (looks like »).

There are also three shorthands for the Cyrillic dash (тире), which is shorter than the emdash but longer than the endash (namely 0.8 em). Note that, since it is not covered by unicode, this character is faked by telescoping two endashes:

`"---` Cyrillic dash for the use in normal text. This requires preceding space in input (trailing space is optional) and prints with a non-breakable thin space before and after the dash.

`"--~` Cyrillic dash for the use in compound names (surnames). As opposed to `"---` this removes any space before and after the dash.

`"--*` Cyrillic dash for denoting direct speech. This adds a larger space after the dash. Space before the dash is output as is.

- **`numerals`** = *`arabic`*, `cyrillic-alph` or `cyrillic-trad`
  Uses either Arabic numerals or Cyrillic alphanumerical numbering. The two Cyrillic variants differ as follows:
  - `cyrillic-alph` steps through the Cyrillic alphabet. Thus it can only be used up to 30.
  - `cyrillic-trad` (= `cyrillic`) uses a traditional Cyrillic alphanumeric system.[7] It supports numbers up to 999 999.
- **`spelling`** = *`modern`* or `classic` (= `tarask`)
  With `spelling=classic`, captions and dates adhere to the Taraškievica (or Belarusian classical) orthography rather than the standard orthography.

Commands:

`\Asbuk`
- `\Asbuk`: produces uppercased Cyrillic alphanumerals, for environments such as `enumerate`. It steps through the Cyrillic alphabet and thus it can only be used up to 30. The command takes a counter as argument, *e.g.*, `\textbelarusian{\Asbuk{page}}` produces Ф.

`\asbuk`
- `\asbuk`: same as `\Asbuk` but in lowercase.

`\AsbukTrad`
- `\AsbukTrad`: same as `\Asbuk` but using the traditional Cyrillic alphanumeric numbering which supports numbers up to 999 999. *E.g.*, `\textbelarusian{\AsbukTrad{page}}` produces КА.

`\asbukTrad`
- `\asbukTrad`: same as `\AsbukTrad` but in lowercase.

v1.2.0

6.5 bengali ←

Options:

- **`numerals`** = `Western`, `Bengali`, or *`Devanagari`*
- **`changecounternumbering`** = `*true` or *`false`*
  Use specified numerals for headings and page numbers.

6.6 catalan

Options:

v1.1.1
- **`babelshorthands`** ← = `*true` or *`false`*
  Activates the shorthands `"l` and `"L` to type geminated l or L.

Commands:

`\l.l`
`\L.L`
v1.1.1
- `\l.l` and `\L.L` ← can be used to type a geminated l, as in *col·laborar*, similar to babel (the glyph U+00B7 MIDDLE DOT is used as a geminating sign).

6.7 croatian

Options:

v1.47
- **`babelshorthands`** ← = `*true` or *`false`*
  If this is turned on, the following shorthands for fine-tuning hyphenation and micro-typography of Croatian words are activated.

`"|`	disables a ligature at this position.

[7]See https://en.wikipedia.org/wiki/Cyrillic_numerals.

`"=`	for an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.
`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).
`""`	allows for a line break at this position (without hyphenation sign).
`"/`	a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

Furthermore, the following shorthands generate easy access to Croatian digraphs (ligatures):

`"dz`	Generates the ligature dž if the font provides it. If not, the two characters are output separately. Also available for `"Dz` (Dž) and `"DZ` (DŽ).
`"lj`	Generates the ligature lj if the font provides it. If not, the two characters are output separately. Also available for `"Lj` (Lj) and `"LJ` (LJ).
`"nj`	Generates the ligature nj if the font provides it. If not, the two characters are output separately. Also available for `"Nj` (Nj) and `"NJ` (NJ).

Finally, there are also four shorthands for quotation marks:

``"` ``	for Croatian left double quotes („).
`"'`	for Croatian right double quotes (”).
`">`	for Croatian left guillemets (»).
`"<`	for Croatian right guillemets («).

v1.47 ▸ **`disableligatures`** ⟵ `=` `*true` or *`false`*

If this is `true`, all Croatian ligatures (for digraphs such as dž) will be replaced by single characters. This can be useful if the ligatures on your font are broken (if the font does not have them, they are automatically replaced).

6.8 czech

Options:

v1.45
- **`babelshorthands`** ⟵ = `*true` or *`false`*
  If this is turned on, the following shorthands for Czech are activated:

`"=`	for an explicit hyphen sign which is repeated at the beginning of the next line when hyphenated, as common in Czech typesetting (only needed with `splithyphens=false`).
`"‘`	for Czech left double quotes („).
`"’`	for Czech right double quotes (“).
`">`	for Czech left double guillemets (»).
`"<`	for Czech right double guillemets («).

v1.45
- **`splithyphens`** ⟵ = `*`*`true`* or `false`
  According to Czech typesetting conventions, if a word with a hard hyphen (such as *je-li*) is hyphenated at this hyphen, a second hyphenation character is to be inserted at the beginning of the line that follows the hyphenation (*je-/-li*). By default, this is done automatically ⟵ (if you are using LuaTEX, the luavlna package is loaded to achieve this). Set this option to `false` to disable the feature. v1.46

v1.45
- **`vlna`** ⟵ = `*`*`true`* or `false`
  According to Czech typesetting conventions, single-letter words (non-syllable prepositions) must not occur at line ends. Polyglossia takes care of this automatically by default ⟵ (if you are using LuaTEX, the luavlna package is loaded to achieve this). Set this option to `false` to disable the feature. v1.46

6.9 dutch

Options:

v1.1.1
- **`babelshorthands`** ⟵ = `*true` or *`false`*
  If this is turned on, the following shorthands defined for fine-tuning hyphenation and micro-typography of Dutch words are activated:

`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-` in default TEX).
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

`"|` disables a ligature at this position.

`""` allows for a line break at this position (without hyphenation sign).

`"/` a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

`\-` In addition, the macro `\-` is redefined to allow hyphens in the rest of the word (equivalent to `"-`).

6.10 english

Options:

- **`variant`** = *`american`* (= `us`), `usmax` (same as `american` but with additional hyphenation patterns), `british` (= `uk`), `australian`, `canadian` ⟵, or `newzealand` v1.45
- **`ordinalmonthday`** = `*true` or *`false`*
  The default value is true for `variant=british`.

6.11 esperanto

Commands:

`\hodiau` `\hodiaun`

- `\hodiau` and `\hodiaun` are special forms of `\today`. The former produces the date in Esperanto preceded by the article (*la*), which is the most common date format. The latter produces the same date format in accusative case.

6.12 finnish

Options:

- **`babelshorthands`** ⟵ = `*true` or *`false`* v1.45
  If this is turned on, the following shorthands for fine-tuning hyphenation and micro-typography of Finnish words are activated:

  `"-` adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).

  `"~` for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

  `"|` disables a ligature at this position.

  `""` allows for a line break at this position (without hyphenation sign).

`"/` a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

6.13 french

Options:

- **`variant`** = *`french`* or `canadian` (= `acadian`) ⟵, `swiss` ⟵ v1.45 v1.47
  Currently, the only difference between the four variants is that `swiss` uses `thincolonspace=true` by default since this conforms to the Swiss conventions.
- **`autospacing`** = **`true`* or `false`
  One of the most distinct features of French typography is the addition of extra spacing around punctuation and quotation marks (guillemets). By default, polyglossia adds these spaces automatically, so you don't need to enter them. This options allows you to switch this feature off globally.
- **`thincolonspace`** ⟵ = *`true` or *`false`* v1.46
  With `variant=swiss`, the default value is `true`. If `false`, a full (non-breaking) interword space is inserted before a colon. If `true`, a thinner space – as before `;`, `!`, and `?` – is used. Note that this option must be set after the `variant` option.
- **`autospaceguillemets`**[8] = **`true`* or `false`
  If you only want to disable the automatic addition of spacing after opening and before closing guillemets (and not at punctuation), set this to *false*. Note that the more general option *autospacing* overrides this.
- **`autospacetypewriter`**[9] ⟵ = *`true` or *`false`* v1.45
  By default, automatic spacing is disabled in typewriter font. If this is enabled, spacing in typewriter context is the same as with roman and sans serif font, depending on the `autospacing` and `autospaceguillemets` settings (note that this was the default up to v. 1.44).
- **`frenchfootnote`** = *`true` or *`false`*
  If `true`, footnotes start with a non-superscripted number followed by a dot, as common in French typography. Note that this might interfere with the specific footnote handling of classes or packages. Also note that this option is only functional (by design) if French is the main language.

[8] Up to version 1.44, the option was called `automaticspacesaroundguillemets`. For backwards compatibility reasons, the more verbose old option is still supported.

[9] Babel's syntax `OriginalTypewriter` is also supported.

v1.46
- **frenchitemlabels** ← = *true or *false*
  If *true*, itemize item labels use em-dashes throughout, as common in French typography. Note that this option is only functional (by design) if French is the main language. Also, it might interfere with list packages such as enumitem.

v1.46
- **itemlabels** ← = ⟨code⟩ (default value: `\textemdash`)
  If *frenchitemlabels* is true, you can customize here the used item label of all levels.

v1.46
- **itemlabeli** ← = ⟨code⟩ (default value: `\textemdash`)
  If *frenchitemlabels* is true, you can customize here the used item label of the first level.

v1.46
- **itemlabelii** ← = ⟨code⟩ (default value: `\textemdash`)
  If *frenchitemlabels* is true, you can customize here the used item label of the second level.

v1.46
- **itemlabeliii** ← = ⟨code⟩ (default value: `\textemdash`)
  If *frenchitemlabels* is true, you can customize here the used item label of the third level.

v1.46
- **itemlabeliv** ← = ⟨code⟩ (default value: `\textemdash`)
  If *frenchitemlabels* is true, you can customize here the used item label of the fourth level.

Commands:

`\NoAutoSpacing` v1.45
- `\NoAutoSpacing` ← disables automatic spacing around punctuation and quotation marks in all following text. The command can also be used locally if braces are used for grouping: `{\NoAutoSpacing foo:bar}`

`\AutoSpacing` v1.45
- `\AutoSpacing` ← enables automatic spacing around punctuation and quotation marks in all following text. The command can also be used locally if braces are used for grouping: `{\AutoSpacing regarde!}`

v1.45
6.14 gaelic ←

Options:
- **variant** = *irish* or scottish

v1.46
6.15 georgian ←

Options:
- **babelshorthands** = *true or *false*
  If this is turned on, the following shorthands are activated:

`"-` adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).

`"=` adds an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).

`"~` for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

`"|` disables a ligature at this position.

`""` allows for a line break at this position (without hyphenation sign).

`",` thinspace for initials with a breakpoint in following surname.

`"‘` for German-style left double quotes (looks like „).

`"’` for German-style right double quotes (looks like “).

`"<` for French-style left double quotes (looks like «).

`">` for French-style right double quotes (looks like »).

There are also three shorthands for the Cyrillic dash (тире), which is shorter than the emdash but longer than the endash (namely 0.8 em). Note that, since it is not covered by unicode, this character is faked by telescoping two endashes:

`"---` Cyrillic dash for the use in normal text. This requires preceding space in input (trailing space is optional) and prints with a non-breakable thin space before and after the dash.

`"--~` Cyrillic dash for the use in compound names (surnames). As opposed to `"---` this removes any space before and after the dash.

`"--*` Cyrillic dash for denoting direct speech. This adds a larger space after the dash. Space before the dash is output as is.

- **`numerals`** = *`arabic`* or `georgian`
  Uses either Arabic numerals or Georgian alphanumerical numbering.
- **`oldmonthnames`** = `*true` or *`false`*
  Uses traditional Georgian month names.

6.16 german

Options:

v1.33.4
- **`variant`** = *`german`*, austrian, or swiss ←
  Setting `variant=austrian` or `variant=swiss` uses some lexical variants. With `spelling=old`, `variant=swiss` furthermore loads specific hyphenation patterns.
- **`spelling`** = *`new`* (= `1996`) or `old` (= `1901`)
  Indicates whether hyphenation patterns for traditional (1901) or reformed (1996) orthography should be used. The latter is the default.

v1.0.3
- **`babelshorthands`** ← = `*true` or *`false`*
  If this is turned on, all shorthands defined in babel for fine-tuning hyphenation and micro-typography of German words are activated.

`"ck`	for `ck` to be hyphenated as `k-k` (1901 spelling).
`"ff`	for `ff` to be hyphenated as `ff-f` (1901 spelling); this is also available for the letters l, m, n, p, r and t.
`"|`	disables a ligature at this position (*e.g.,* `Auf"|lage`).
`"=`	for an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable, *e.g.,* `bergauf und "~ab`.
`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).
`""`	allows for a line break at this position (without hyphenation sign); *e.g.,* `(pseudo"~)""wissenschaftlich`.
`"/`	a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

There are also four shorthands for quotation signs:

`` "` ``	for German-style left double quotes („)
`"'`	for German-style right double quotes (“)
`"<`	for French-style left double quotes («)
`">`	for French-style right double quotes (»).

v1.2.0 v1.46

- **script** ← = *latin* or blackletter ← (= fraktur)
  Setting script=blackletter adapts the captions for typesetting German in blackletter type (using the long s (ſ) where appropriate).

6.17 greek

Options:

- **variant** = *monotonic* (= mono), polytonic (= poly), or ancient
- **numerals** = *greek* or arabic
- **attic** = *true or *false*

Commands:

\Greeknumber \greeknumber \atticnumeral \atticnum

- \Greeknumber and \greeknumber (see section 8.3).
- The command \atticnumeral (= \atticnum) (activated with the option attic=true), displays numbers using the acrophonic numbering system (defined in the Unicode range U+10140–U+10174).[10]

6.18 hebrew

Options:

- **numerals** = hebrew or *arabic*
- **calendar** = hebrew or *gregorian*
- **marcheshvan** = *true or *false*
  If true, the second month of the civil year will be output as מרחשון (Marcheshvan) rather than חשון (Heshvan), which is the default.

Commands:

\hebrewnumeral \hebrewalph \aemph

- \hebrewnumeral (= \hebrewalph) (see section 8.3).
- \aemph (see section 6.2).

v1.2.0

6.19 hindi ←

Options:

- **numerals** = Western or *Devanagari*

[10]See the documentation of the xgreek package for more details.

6.20 hungarian

Options:

v1.46

- **`swapstrings`** ⇠ = **`all`*, `captions`, `headings`, `headers`, `hheaders`, or `none`
  In Hungarian, some caption strings need to be in a different order than in other languages (*e.g., 1. fejezet* instead of *Chapter 1*). By default, polyglossia tries hard to provide the correct order for different classes and packages (standard classes, KOMA-script, memoir, and titlesec package should work, as well as fancyhdr and caption). However, since the definition of these strings is not standardized, the redefinitions might not work and even interfere badly if you use specific classes or packages that redefine the respective strings themselves. In this case, you can disable some or all changes. The possibilities are:
  - `all`: Redefine figure and table captions, part and chapter headings, and running headers (= default setting)
  - `captions`: Redefine figure and table captions only
  - `headings`: Redefine part and chapter headings only
  - `headers`: Redefine running headers only
  - `hheaders`: Redefine part and chapter headings as well as running headers
  - `none`: Do not redefine anything

Commands:

`\ontoday`
`\ondatehungarian`

- `\ontoday` (= `\ondatehungarian`): special form of `\today` which produces a slightly different date format as used in prepositional phrases (such as 'on February 10th') in Hungarian.

6.21 italian

Options:

v1.2.0cc

- **`babelshorthands`** ⇠ = *`true` or *`false`*
  Activates the `"` character as a switch to perform etymological hyphenation when followed by a letter. Furthermore, the following shorthands are activated:

`""`	double raised open quotes (the Italian keyboard misses the backtick).
`"<`	open guillemet (looks like «).
`">`	closing guillemet (looks like »).

`"/` a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.

`"-` adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).

6.22 korean ← v1.40.0

Options:

- **`variant`** = *`plain`*, `classic`, or `modern`
  These variants control spacing before/after CJK punctuations.
  - `plain`: Do nothing
  - `classic`: Suitable for text with no interword spaces. This option forces CJK punctuations to half-width, and inserts half-width glue around them.
  - `modern`: Suitable for text with interword spaces. This option forces CJK punctuations to half-width, and inserts small (half of interword space) glue around them.
- **`captions`** = *`hangul`* or `hanja`
- **`swapstrings`** ← = **`all`*, `headers`, `headings`, or `none` v1.47
  With this option, Korean-style part and chapter headings, and running headers are available. It is similar to Hungarian (see 6.20) except that figure and table captions are not touched.
  - `all`: Redefine part and chapter headings, and running headers (= default setting)
  - `headings`: Redefine part and chapter headings only
  - `headers`: Redefine running headers only
  - `none`: Do not redefine anything

6.23 kurdish ← v1.45

Options:

- **`variant`** = `kurmanji` or *`sorani`*
- **`script`** = `Arabic` or `Latin`
  Defaults are `Arabic` for Sorani and `Latin` for Kurmanji.
- **`numerals`** = `western` or `eastern`
  Defaults are `western` for Latin and `eastern` for Arabic script, depending on the selection above.

- **abjadjimnotail** = *true or *false*
  Set this to true if you want the *abjad* form of the number three to be ﺟ – as in the manuscript tradition – instead of the modern usage ج.

Commands:

\ontoday
- \ontoday: special form of \today which produces a slightly different date format as used in prepositional phrases (as in 'on February 10th'). Only available for Latin script.

\abjad
- \abjad (see section 8.3)

\aemph
- \aemph (see section 6.2)

v1.2.0

6.24 lao ←

Options:

- **numerals** = lao or *arabic*

6.25 latin

Options:

v1.46
- **variant** = classic, medieval, *modern*, or ecclesiastic ←
  These variants refer to different spelling conventions. The classic and the medieval variant do not use the letters *U* and *v*, but only *V* and *u*. This concerns predefined terms like month names as well as the behaviour of the \MakeUppercase and the \MakeLowercase command. The medieval and the ecclesiastic variant use the ligatures *æ* and *œ*. See table 4 for examples.
  Furthermore, the ecclesiastic variant takes care for a punctuation spacing similar to French, but with smaller spaces, as provided for PDFTEX by the ecclesiastic package.

v1.46
- **hyphenation** ← = classic, modern, or liturgical
  There are three different sets of hyphenation patterns for Latin. Separate documention for them is available on the Internet.[11] Each of the four variants mentioned above has its default set of hyphenation patterns as indicated by table 5. Use the hyphenation option if the default style does not fit your needs. Note that the liturgical hyphenation patterns are the

[11] https://github.com/gregorio-project/hyphen-la/blob/master/doc/README.md#hyphenation-styles

Table 4. Spelling differences between the Latin language variants. The capitalization of month names and the use of *i/j* may be affected by the `capitalizemonth` and the `usej` option.

classic	**medieval**	**modern**	**ecclesiastic**
Ianuarii	Ianuarii	Ianuarii	ianuarii
Nouembris	Nouembris	Novembris	novembris
Praefatio	Præfatio	Praefatio	Præfatio
`\MakeUppercase{Iulius}` yields:			
IVLIVS	IVLIVS	IULIUS	IULIUS

Table 5. Latin default hyphenation styles

Language variant	**Default hyphenation style**
classic	classic
medieval	modern
modern	modern
ecclesiastic	modern

default of none of the language variants. To use them, you have to say `hyphenation=liturgical` in any case.

v1.46 ▸ **`ecclesiasticfootnotes`** ⟵ = *`true` or *`false`*
Use footnotes as provided by the ecclesiastic package, which typesets footnotes with ordinary instead of superior numbers and without indentation. As many ecclesiastic documents and liturgical books use footnotes that are very similar to the ordinary LaTeX ones, we do not use this footnote style as default even for the `ecclesiastic` variant.
Note that this option is only possible if Latin is the main language of your document.

v1.46 ▸ **`usej`** ⟵ = *`true` or *`false`*
Use *J/j* in predefined terms. The letter *j* is not of ancient origin. In early modern times, it was used to distinguish the consonantic *i* from the vocalic *i*. Nowadays, the use of *j* has disappeared from most Latin publications. So `false` is the default value for all four language variants. Use this option if you prefer *Januarii* and *Maji* to *Ianuarii* and *Maii*.

v1.46 ▸ **`capitalizemonth`** ⟵ = *`true` or `false`

Capitalize the month name when printing dates (using the `\today` command). Traditionally, month names are capitalized. However, in recent liturgical books they are lowercase. So `true` is the default value for the variants `classic`, `medieval`, and `modern`, whereas `false` is the default value for the `ecclesiastic` variant.

▸ **`babelshorthands`** = `*true` or *`false`*
Enable the following shorthands inherited from babel-latin and the ecclesiastic package.

`"<`	for « (left guillemet)
`">`	for » (right guillemet)
`"`	If no other shorthand applies, `"` before any letter character defines an optional break point allowing further break points within the same word (as opposed to the `\-` command).
`"|`	the same as `"`, but also possible before non-letter characters
`'a`	for á (a with acute), also available for é, í, ó, ú, ý, ǽ, and œ́
`'A`	for Á (A with acute), also available for É, Í, Ó, Ú, V́, Ý, Ǽ, and Œ́

The following shorthands are only available for the `medieval` and the `ecclesiastic` variant.

`"ae`	for æ (ae ligature), also available for œ
`"Ae`	for Æ (AE ligature), also available for Œ
`"AE`	for Æ (AE ligature), also available for Œ
`'ae`	for ǽ (ae ligature with acute), also available for œ́
`'Ae`	for Ǽ (AE ligature with acute), also available for Œ́
`'AE`	for Ǽ (AE ligature with acute), also available for Œ́

v1.46 ▸ **`prosodicshorthands`** ⟵ = `*true` or *`false`*
Enable shorthands for prosodic marks (macrons and breves) very similiar to those provided by babel-latin using the `withprosodicmarks` modifier. Note that the active `=` character used for macrons will cause problems with commands using `key=value` interfaces, such as the command `\includegraphics[scale=2]{...}`. Use `\shorthandoff{=}` before such commands (and `\shorthandon{=}` thereafter) within every environment with prosodic shorthands enabled.
The following shorthands are available.

`=a`	for ā (a with macron), also available for ē, ī, ō, ū, and ȳ

`=A` for Ā (A with macron), also available for Ē, Ī, Ō, Ū, V̄, and Ȳ. Note that a macron above the letter V is only displayed if your font supports the Unicode character `0304` (*combining macron*).

`=ae` for a͞e (ae diphthong with macron), also available for a͞u, e͞u, and o͞e. Note that macrons above diphthongs are only displayed if your font supports the Unicode character `035E` (*combining double macron*).

`=Ae` for A͞e (Ae diphthong with macron), also available for A͞u, E͞u, and O͞e.

`=AE` for A͞E (AE diphthong with macron), also available for A͞U, E͞U, and O͞E.

`^a` for ă (a with breve), also available for ĕ, ĭ, ŏ, ŭ, and y̆. Note that a breve above the letter y is only displayed if your font supports the Unicode character `0306` (*combining breve*).

`^A` Ă (A with breve), also available for Ĕ, Ĭ, Ŏ, Ŭ, V̆, and Y̆. Note that breves above the letters V and Y are only displayed if your font supports the Unicode character `0306` (*combining breve*).

6.26 malay

Options:

- v1.45 **`variant`** ⟵ = `*indonesian*` or `malaysian`

6.27 marathi

Options:

- **`numerals`** = `*Devanagari*` or `Western`

6.28 mongolian ⟵

v1.45

Currently, only the Khalkha variety in Cyrillic script is supported.

Options:

- **`babelshorthands`** = `*true` or `*false*`

  If this is turned on, the following shorthands are activated:

  `"-` adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).

`"=` adds an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).

`"~` for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

`"|` disables a ligature at this position.

`""` allows for a line break at this position (without hyphenation sign).

`",` thinspace for initials with a breakpoint in following surname.

`"‘` for German-style left double quotes (looks like „).

`"’` for German-style right double quotes (looks like “).

`"<` for French-style left double quotes (looks like «).

`">` for French-style right double quotes (looks like »).

There are also three shorthands for the Cyrillic dash (тире), which is shorter than the emdash but longer than the endash (namely 0.8 em). Note that, since it is not covered by unicode, this character is faked by telescoping two endashes:

`"---` Cyrillic dash for the use in normal text. This requires preceding space in input (trailing space is optional) and prints with a non-breakable thin space before and after the dash.

`"--~` Cyrillic dash for the use in compound names (surnames). As opposed to `"---` this removes any space before and after the dash.

`"--*` Cyrillic dash for denoting direct speech. This adds a larger space after the dash. Space before the dash is output as is.

- **`numerals`** = *`arabic`*, `cyrillic-alph` or `cyrillic-trad`
  Uses either Arabic numerals or Cyrillic alphanumerical numbering. The two Cyrillic variants differ as follows:
  - `cyrillic-alph` steps through the Cyrillic alphabet. Thus it can only be used up to 30.
  - `cyrillic-trad` (= `cyrillic`) uses a traditional Cyrillic alphanumeric system.[12] It supports numbers up to 999 999.

Commands:

`\Asbuk`

- `\Asbuk`: produces uppercased Cyrillic alphanumerals, for environments

[12] See `https://en.wikipedia.org/wiki/Cyrillic_numerals`.

such as `enumerate`. It steps through the Cyrillic alphabet and thus it can only be used up to 30. The command takes a counter as argument, *e.g.*, `\textmongolian{\Asbuk{section}}` produces Е.

`\asbuk`
- `\asbuk`: same as `\Asbuk` but in lowercase.

`\AsbukTrad`
- `\AsbukTrad`: same as `\Asbuk` but using the traditional Cyrillic alphanumeric numbering which supports numbers up to 999 999.
  E.g., `\textmongolian{\AsbukTrad{section}}` produces Ѕ.

`\asbukTrad`
- `\asbukTrad`: same as `\AsbukTrad` but in lowercase.

6.29 norwegian

Options:

v1.45
- **`variant`** ⟵ = `bokmal` or *`nynorsk`*

6.30 persian

Options:

- **`numerals`** = `western` or *`eastern`*

v1.0.3
- **`abjadjimnotail`** ⟵ = `*true` or *`false`*
  Set this to `true` if you want the *abjad* form of the number three to be ﺟ – as in the manuscript tradition – instead of the modern usage ج.

Commands:

`\abjad`
- `\abjad` (see section 8.3)

`\aemph`
- `\aemph` (see section 6.2).

6.31 portuguese

Options:

v1.45
- **`variant`** ⟵ = `brazilian` or *`portuguese`*

6.32 russian

Options:

- **`babelshorthands`** = `*true` or *`false`*
  If this is turned on, the following shorthands are activated:

  `"-` adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).

`"=` adds an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).

`"~` adds an explicit hyphen without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.

`"|` disables a ligature at this position.

`""` allows for a line break at this position (without hyphenation sign).

There are also three shorthands for the Cyrillic dash (тире), which is shorter than the emdash but longer than the endash (namely 0.8 em). Note that, since it is not covered by unicode, this character is faked by telescoping two endashes:

`"---` Cyrillic dash for the use in normal text. This requires preceding space in input (trailing space is optional) and prints with a non-breakable thin space before and after the dash.

`"--~` Cyrillic dash for the use in compound names (surnames). As opposed to `"---` this removes any space before and after the dash.

`"--*` Cyrillic dash for denoting direct speech. This adds a larger space after the dash. Space before the dash is output as is.

v1.50

- **`forceheadingpunctuation`** ⟵ = **true* or `false`
  By default, chapter and section numbers always have a trailing punctuation in Russian (as in *1.1.* as opposed to *1.1*). If this option is set to false, polyglossia will not touch heading punctuation, so this will be whatever the class or a package determines.

v1.46

- **`indentfirst`** ⟵ = **true* or `false`
  By default, all paragraphs are indented in Russian, also those after a chapter or section heading. If this option is false, the latter paragraphs are not indented, as normal in LaTeX.
- **`spelling`** = *modern* or `old`
  This option is for captions and date only, not for hyphenation.
- **`numerals`** = *arabic*, `cyrillic-alph` or `cyrillic-trad`
  Uses either Arabic numerals or Cyrillic alphanumerical numbering. The two Cyrillic variants differ as follows:
  - `cyrillic-alph` steps through the Cyrillic alphabet. Thus it can only be used up to 30.
  - `cyrillic-trad` (= `cyrillic`) uses a traditional Cyrillic alphanumeric

system.[13] It supports numbers up to 999 999.

Commands:

`\Asbuk`
- `\Asbuk`: produces uppercased Cyrillic alphanumerals, for environments such as `enumerate`. It steps through the Cyrillic alphabet and thus it can only be used up to 30. The command takes a counter as argument, *e.g.*, `\textrussian{\Asbuk{section}}` produces E.

`\asbuk`
- `\asbuk`: same as `\Asbuk` but in lowercase.

`\AsbukTrad`
- `\AsbukTrad`: same as `\Asbuk` but using the traditional Cyrillic alphanumeric numbering which supports numbers up to 999 999.
  E.g., `\textrussian{\AsbukTrad{page}}` produces ЛΘ.

`\asbukTrad`
- `\asbukTrad`: same as `\AsbukTrad` but in lowercase.

v1.45
6.33 sami ←

Currently support for Sami is limited to Northern Sami.

6.34 sanskrit

Options:

v1.0.2
- **`script`** ← = *`Devanagari`*, `Gujarati`, `Malayalam`, `Bengali`, `Kannada`, `Telugu`, or `Latin`
  The value is passed to fontspec in cases where the respective `\⟨script⟩font` is not defined. This can be useful if you typeset Sanskrit texts in scripts other than Devanagari.

v1.45
- **`numerals`** ← = *`Devanagari`* or `Western`

6.35 serbian

Options:

- **`script`** = *`Cyrillic`* or `Latin`
- **`numerals`** = *`arabic`*, `cyrillic-alph` or `cyrillic-trad`
  Uses either Arabic numerals or Cyrillic alphanumerical numbering. The two Cyrillic variants differ as follows:
  - `cyrillic-alph` steps through the Cyrillic alphabet. Thus it can only be used up to 30.

[13] See `https://en.wikipedia.org/wiki/Cyrillic_numerals`.

- `cyrillic-trad` (= `cyrillic`) uses a traditional Cyrillic alphanumeric system.[14] It supports numbers up to 999 999.

Commands:

`\Asbuk`
- `\Asbuk`: produces uppercased Cyrillic alphanumerals, for environments such as `enumerate`. It steps through the Cyrillic alphabet and thus it can only be used up to 30. The command takes a counter as argument, *e.g.*, `\textserbian{\Asbuk{section}}` produces E.

`\asbuk`
- `\asbuk`: same as `\Asbuk` but in lowercase.

`\AsbukTrad`
- `\AsbukTrad`: same as `\Asbuk` but using the traditional Cyrillic alphanumeric numbering which supports numbers up to 999 999.
  E.g., `\textserbian{\AsbukTrad{page}}` produces M.

`\asbukTrad`
- `\asbukTrad`: same as `\AsbukTrad` but in lowercase.

6.36 slovak

Options:

v1.46
- **`babelshorthands`** ⟵ = `*true` or *`false`*
  If this is turned on, the following shorthands for Slovak are activated:

`"=`	for an explicit hyphen sign which is repeated at the beginning of the next line when hyphenated, as common in Slovak typesetting (only needed with `splithyphens=false`).
`"|`	disables a ligature at this position.
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.
`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).
`""`	allows for a line break at this position (without hyphenation sign).
`"/`	a slash that allows for a subsequent line break. As opposed to `\slash`, hyphenation at the breakpoints preset in the hyphenation patterns is still allowed.
`"‘`	for Slovak left double quotes (looks like „).
`"’`	for Slovak right double quotes (looks like “).
`">`	for Slovak left double guillemets (looks like »).

[14]See `https://en.wikipedia.org/wiki/Cyrillic_numerals`.

`"<`   for Slovak right double guillemets (looks like «).

v1.46
- **`splithyphens`** ⟵ = `*`*`true`* or `false`
  According to Slovak typesetting conventions, if a word with a hard hyphen (such as *je-li*) is hyphenated at this hyphen, a second hyphenation character is to be inserted at the beginning of the line that follows the hyphenation (*je-/-li*). By default, this is done automatically (if you are using LuaTEX, the luavlna package is loaded to achieve this). Set this option to `false` to disable the feature.

v1.46
- **`vlna`** ⟵ = `*`*`true`* or `false`
  According to Slovak typesetting conventions, single-letter words (non-syllable prepositions) must not occur at line ends. Polyglossia takes care of this automatically by default (if you are using LuaTEX, the luavlna package is loaded to achieve this). Set this option to `false` to disable the feature.

6.37 slovenian

Options:

- **`localalph`** = `*true` or *`false`*
  If `true`, alpha-numeric counters will use a localized version including characters with caron (a, b, c, č, d, ...).

6.38 sorbian

Options:

v1.45
- **`variant`** ⟵ = `lower` or *`upper`*

v1.45
- **`olddate`** ⟵ = `*true` or *`false`*
  If `true`, `\today` will use traditional Sorbian month names (*i.e.,* it will be synonymous to `\oldtoday` below).

Commands:

`\oldtoday`
- `\oldtoday`: outputs the current date using traditional Sorbian month names, even if `olddate` is `false`.

6.39 spanish

Options:

v1.46
- **`variant`** ⟵ = *`spanish`* or `mexican`

v1.46
- **`spanishoperators`** ⟵ = `*all`, `accented`, `spaced`, `none`, or *`false`*
  Determines of and how math operators are localized to Spanish.

- `accented` causes some math operators to use accents where usual in Spanish (*lím*, *lím sup*, *lím inf*, *máx*, *mín*, *ínf*, *mód*).
- `spaced` causes some math operators to use spaces where usual in Spanish (*arc cos*, *arc sen*, *arc tg*).
- `all` activates `accented` and `spaced` and furthermore provides Spanish localizations of `\sin` (*sen*), `\tan` (*tg*), `\sinh` (*senh*), and `\tanh` (*tgh*).
- `none` does no localization at all (default setting).

v1.46

Commands: ←

- `\arcsen`: alias to `\arcsin` (babel compatibility)
- `\arctg`: alias to `\arctan` (babel compatibility)
- `\sen`: alias to `\sin` (babel compatibility)
- `\senh`: alias to `\sinh` (babel compatibility)
- `\tg`: alias to `\tan` (babel compatibility)
- `\tgh`: alias to `\tanh` (babel compatibility)
- `\spanishoperator`: allows you to define further localized operators. For instance, `\spanishoperator{cotg}` defines a command `\cotg` that outputs *cotg* in math. The optional argument of the command lets you specify the spelling, if needed, *e.g.*, `\spanishoperator[arc\,ctg]{arcctg}`.

6.40 syriac

Options:

v1.0.1

- **`numerals`** ← = *`western`* (*i.e.*, 1234567890), `eastern` (for which the Oriental Arabic numerals are used: ١٢٣٤٥٦٧٨٩٠), or `abjad`

Commands:

- `\abjadsyriac` (see section 8.3)
- `\aemph` (see section 6.2).

6.41 thai

Options:

- **`numerals`** = `thai` or *`arabic`*

To insert word breaks, you need to use an external processor. See the documentation to thai-latex and the file `testthai.tex` that comes with this package.

6.42 tibetan

Options:

- **`numerals`** = `tibetan` or *`arabic`*

6.43 ukrainian

Options:

- **`babelshorthands`** = `*true` or *`false`*

  If this is turned on, the following shorthands are activated:

`"-`	adds a hyphenation point that does still allow for hyphenation at the points preset in the hyphenation patterns (as opposed to `\-`).
`"=`	adds an explicit hyphen with a breakpoint, allowing for hyphenation at the other points preset in the hyphenation patterns (as opposed to plain `-`).
`"~`	for a hyphen sign without a breakpoint. Useful for cases where the hyphen should stick at the following syllable.
`"|`	disables a ligature at this position.
`""`	allows for a line break at this position (without hyphenation sign).

  There are also three shorthands for the Cyrillic dash (тире), which is shorter than the emdash but longer than the endash (namely 0.8 em). Note that, since it is not covered by unicode, this character is faked by telescoping two endashes:

`"---`	Cyrillic dash for the use in normal text. This requires preceding space in input (trailing space is optional) and prints with a non-breakable thin space before and after the dash.
`"--~`	Cyrillic dash for the use in compound names (surnames). As opposed to `"---` this removes any space before and after the dash.
`"--*`	Cyrillic dash for denoting direct speech. This adds a larger space after the dash. Space before the dash is output as is.

- **`numerals`** = *`arabic`*, `cyrillic-alph` or `cyrillic-trad`

  Uses either Arabic numerals or Cyrillic alphanumerical numbering. The two Cyrillic variants differ as follows:

  - `cyrillic-alph` steps through the Cyrillic alphabet. Thus it can only be used up to 30.

- `cyrillic-trad` (= `cyrillic`) uses a traditional Cyrillic alphanumeric system.[15] It supports numbers up to 999 999.

Commands:

`\Asbuk`
- `\Asbuk`: produces uppercased Cyrillic alphanumerals, for environments such as `enumerate`. It steps through the Cyrillic alphabet and thus it can only be used up to 30. The command takes a counter as argument, *e.g.*, `\textukrainian{\Asbuk{section}}` produces Е.

`\asbuk`
- `\asbuk`: same as `\Asbuk` but in lowercase.

`\AsbukTrad`
- `\AsbukTrad`: same as `\Asbuk` but using the traditional Cyrillic alphanumeric numbering which supports numbers up to 999 999.
  E.g., `\textukrainian{\AsbukTrad{page}}` produces МД.

`\asbukTrad`
- `\asbukTrad`: same as `\AsbukTrad` but in lowercase.

6.44 welsh

Options:

- **`date`** = `long` or *`short`*

7 Modifying or extending captions, date formats and language settings

Polyglossia uses the following macros to define language-specific captions (*i.e.*, strings such as "chapter"), date formats and additional language settings (`⟨lang⟩` is to be replaces with the respective language name):

`\captions⟨lang⟩`
- `\captions⟨lang⟩` stores definitions of caption strings (such as, in the case of English, `\def\chaptername{Chapter}`)

`\date⟨lang⟩`
- `\date⟨lang⟩` stores definitions of date formats (usually redefinitions of `\today`, in some cases also definitions of additional date commands)

`\blockextras⟨lang⟩`
- `\blockextras⟨lang⟩` stores macros that are to be executed when the language `⟨lang⟩` is activated via `\selectlanguage`command or the `⟨lang⟩` environment

`\inlineextras⟨lang⟩`
- `\inlineextras⟨lang⟩` stores macros that are to be executed when the language `⟨lang⟩` is activated locally via `\text⟨lang⟩` command

`\noextras⟨lang⟩`
- `\noextras⟨lang⟩` stores macros that are to be executed when the language `⟨lang⟩` is closed

[15]See `https://en.wikipedia.org/wiki/Cyrillic_numerals`.

In order to redefine internal macros, we recommend to use the command `\gappto`. For compatibility with babel the command `\addto` is also available to the same effect. For instance, to change the `\chaptername` for language `lingua`, you can do this:

```
\gappto\captionslingua{\def\chaptername{Caput}}
```

Note that this needs to be done after the respective language has been loaded with `\setmainlanguage` or `\setotherlanguage`.

Specifically for package authors, analogous commands are provided which are only executed if a specific language *variety* is used. As opposed to the macros above, these refer to babel names. Other than that, the function is identical:

`\captions@bbl@⟨babelname⟩` ▸ `\captions@bbl@⟨babelname⟩`
`\date@bbl@⟨babelname⟩` ▸ `\date@bbl@⟨babelname⟩`
`\blockextras@bbl@⟨babelname⟩` ▸ `\blockextras@bbl@⟨babelname⟩`
`\inlineextras@bbl@⟨babelname⟩` ▸ `\inlineextras@bbl@⟨babelname⟩`
`\noextras@bbl@⟨babelname⟩` ▸ `\noextras@bbl@⟨babelname⟩`

By default, these macros are undefined. If they are defined (*e.g.,* by an external package), they will be executed after their respective `⟨lang⟩` counterpart and thus can be used to overwrite definitions of the former. Again, use `\gappto` to define/modify these macros. For instance, to add a new caption `\footnotename` to the Swiss variety of German (babel name `nswissgerman`), you can do this:

```
\gappto\captions@bbl@nswissgerman{\def\footnotename{Fussnote}}
```

If you do this in a document preamble rather than in a package, you need to embrace the redefinition by `\makeatletter` and `\makeatother` due to the `@` in the macro names.

Finally, as soon as the language has been switched (either inline or as a block), polyglossia executes the (by default empty) hook

`\polyglossia@language@switched` ▸ `\polyglossia@language@switched`

to which you can append arbitrary code (via `\gappto`) that should be executed if (a particular) language is being activated. This is done before any of the above macros are issued (so you can still alter them), but at a point where `\languagename`, `\babelname` and `\languageid` are already set, so you can condition on specific languages in your code. This hook is particularly provided for package authors.

8 Script-specific numbering

Languages and scripts have specific numbering conventions. Some use decimal digits (*e.g.,* Arabic numerals), some use alphabetic or alphanumerical notation (*e.g.,* Roman numbering). In some cases, different conventions are available (*e.g.,* Mashriq or Maghrib numbering in Arabic script, Arabic or Hebrew [= alphanumeric] numbering in Hebrew).

If the latter is the case, polyglossia provides language options which allow you to select or switch to the suitable convention. With the appropriate language option set, polyglossia will automatically convert the output of internal LaTeX counters to their localized forms, for instance to display page, chapter and section numbers.

For manual input of numbers, macros are provided. These convert Arabic numeric input to the respective local decimal digit (see sec. 8.2), alphanumeric representation (see sec. 8.3) or whatever is appropriate (see sec. 8.1). The possibilities are described in turn.

8.1 General localization of numbering

v1.45 `\localnumeral` As of 1.45, ← polyglossia provides a generic macro `\localnumeral` which converts numbers to the current local form (which might be script-specific decimal digit, an alphabetic numbering or something else). For instance in an Arabic environment `\localnumeral{42}` yields ٤٢, whereas in an Hebrew environment, it results in מב with `numerals=hebrew`, and 42 with `numerals=arabic`. Note that, as opposed to the various `digits` macros (described in sec. 8.2), the argument of `\localnumeral` must consist of numbers only.

v1.45 `\localnumeral*` For ← the conversion of counters, the starred version `\localnumeral*` is provided. This takes a counter as argument. For instance in an Arabic environment `\localnumeral*{page}` yields ٤٦.

`\Localnumeral` `\Localnumeral*` For scripts with alphanumeric numbering, the variants `\Localnumeral` and `\Localnumeral*` provide the uppercased versions.

All these macros provide the following options:

`[lang=]`
- **`lang`** = *`local`*, `main`, or ⟨`language`⟩
  Output number in the local form of the currently active language for `local`, the main language of the document for `main`, and any (loaded) language for ⟨`language`⟩ (*e.g.,* `\localnumeral[lang=arabic]{42}`).

8.2 Non-Western decimal digits

v1.1.1 In addition ← to the generic macros described above, polyglossia provides language-specific conversion macros which can be used if the generic ones do not suit the need.[16] The macros have the form `\⟨script⟩digits`. They convert Arabic numerical input and leave every other input untouched. In an Arabic context, for instance, `\arabicdigits{9182/738543-X}` yields ٩١٨٢/٧٣٨٥٤٣-X.

Currently, the following macros are provided:

`\arabicdigits` ▸ `\arabicdigits`

`\bengalidigits` ▸ `\bengalidigits`

`\devanagaridigits` ▸ `\devanagaridigits`

`\farsidigits` ▸ `\farsidigits`

`\kannadadigits` ▸ `\kannadadigits`

`\khmerdigits` ▸ `\khmerdigits`

`\laodigits` ▸ `\laodigits`

`\nkodigits` ▸ `\nkodigits`

`\thaidigits` ▸ `\thaidigits`

`\tibetandigits` ▸ `\tibetandigits`

8.3 Non-Latin alphabetic numbering

For languages which use special (non-Latin) alphanumerical notation[17], dedicated macros are provided.

They work in a similar way than the `\⟨script⟩digits` macros described above: They take Arabic numerical input and output the respective value in the local alphabetic numbering scheme (most of these macros are equivalent to `\localnumeral` and `\Localnumeral` in the respective context).

The following macros are provided:

`\abjad` ▸ `\abjad` outputs Arabic *abjad* numbers according to the Mashriq varieties. Example: `\abjad{1863}` yields غضسج.

`\abjadmaghribi` ▸ `\abjadmaghribi` outputs Arabic *abjad* numbers according to the Maghrib varieties. Example: `\abjadmaghribi{1863}` yields شظصج.

[16] A third method are so-called TECKit fontmappings. Those can be activated with the fontspec `Mapping` option, using `arabicdigits`, `farsidigits` or `thaidigits`. For instance if `\arabicfont` is defined with the option `Mapping=arabicdigits`, typing `\textarabic{2010}` results in ٢٠١٠. Note that this method has some drawbacks, though, for instance when the value of a counter has to be written and read from auxiliary files. So please use this with care.

[17] For instance, see http://en.wikipedia.org/wiki/Greek_numerals, http://en.wikipedia.org/wiki/Abjad_numerals, http://en.wikipedia.org/wiki/Hebrew_numerals, and http://en.wikipedia.org/wiki/Syriac_alphabet.

`\abjadsyriac`

▸ `\abjadsyriac` outputs Syriac abjad numerals.[18]
Example: `\abjadsyriac{463}` yields ܬܣܓ.

`\armeniannumeral`

▸ `\armeniannumeral` produces Armenian alphabetic numbering. Example: `\armeniannumeral{1863}` yields ՌՊԿԳ.

`\belarusiannumeral`
`\Belarusiannumeral`

▸ `\belarusiannumeral` produces Belarusian numbering, with uppercased variant (for alphanumerical variant) via `\Belarusiannumeral`. Depending on the `numerals` option in the Belarusian language selection, this is either Arabic digit or Cyrillic alphanumercial output.
Example: With `numerals=latin` `\belarusiannumeral{19}` yields 19, with `numerals=cyrillic-trad` `\belarusiannumeral{19}` results in іѳ,
with `numerals=cyrillic-alph` `\belarusiannumeral{19}` results in у.

`\georgiannumeral`

▸ `\georgiannumeral` produces Georgian alphabetic numbering.
Example: `\georgiannumeral{1863}` yields ჩყჲგ.

`\greeknumeral`
`\Greeknumeral`

▸ `\greeknumeral` produces Greek alphabetic numbering, `\Greeknumeral` outputs uppercased variants. Example: `\greeknumeral{1863}` yields ͵αωξγʹ, `\Greeknumeral{1863}` results in ͵ΑΩΞΓʹ.

`\hebrewnumeral`
`\Hebrewnumeral`
`\Hebrewnumeralfinal`

▸ `\hebrewnumeral`, `\Hebrewnumeral` and `\Hebrewnumeralfinal` generate variants of Hebrew alphanumeric numerals. The commands behave exactly as they do in babel: `\hebrewnumeral` outputs the numbers without any decoration, `\Hebrewnumeral` adds *gereshayim* before the last letter, `\Hebrewnumeralfinal` uses in addition the final forms of Hebrew letters. Examples: `\hebrewnumeral{1750}` yields א׳תשנ, `\Hebrewnumeral{1750}` yields א׳תש״נ, and `\Hebrewnumeralfinal{1750}` yields א׳תש״ן.

`\mongoliannumeral`
`\Mongoliannumeral`

▸ `\mongoliannumeral` produces Mongolian numbering, with uppercased variant (for alphanumerical variant) via `\Mongoliannumeral`. Depending on the `numerals` option in the Mongolian language selection, this is either Arabic digit or Cyrillic alphanumercial output.
Example: With `numerals=latin` `\mongoliannumeral{19}` yields 19, with `numerals=cyrillic-trad` `\mongoliannumeral{19}` results in іѳ,
with `numerals=cyrillic-alph` `\mongoliannumeral{19}` results in у.

`\russiannumeral`
`\Russiannumeral`

▸ `\russiannumeral` produces Russian numbering, with uppercased variant (for alphanumerical variant) via `\Russiannumeral`. Depending on the `numerals` option in the Russian language selection, this is either Arabic digit or Cyrillic alphanumercial output.
Example: With `numerals=latin` `\russiannumeral{19}` yields 19, with

[18] A fine guide to numerals in Syriac can be found at http://www.garzo.co.uk/documents/syriac-numerals.pdf.

`numerals=cyrillic-trad` `\russiannumeral{19}` results in iѳ,
with `numerals=cyrillic-alph` `\russiannumeral{19}` results in у.

`\serbiannumeral`
`\Serbiannumeral`

- `\serbiannumeral` produces Serbian numbering, with uppercased variant (for alphanumerical variant) via `\Serbiannumeral`. Depending on the `numerals` option in the Serbian language selection, this is either Arabic digit or Cyrillic alphanumercial output.
  Example: With `numerals=latin` `\serbiannumeral{19}` yields 19, with `numerals=cyrillic-trad` `\serbiannumeral{19}` results in iѳ,
  with `numerals=cyrillic-alph` `\serbiannumeral{19}` results in у.

`\ukrainiannumeral`
`\Ukrainiannumeral`

- `\ukrainiannumeral` produces Ukrainian numbering, with uppercased variant (for alphanumerical variant) via `\Ukrainiannumeral`. Depending on the `numerals` option in the Ukrainian language selection, this is either Arabic digit or Cyrillic alphanumercial output.
  Example: With `numerals=latin` `\ukrainiannumeral{19}` yields 19, with `numerals=cyrillic-trad` `\ukrainiannumeral{19}` results in iѳ,
  with `numerals=cyrillic-alph` `\ukrainiannumeral{19}` results in у.

9 Footnotes in right-to-left context

With languages that use right-to-left scripts, footnote apparatuses are usually placed at the right side of the page bottom. Consequently, the footnote rule also is to be placed right. Things get more tricky, though, if right-to-left and left-to-right scripts are mixed. Then you might want to put the footnotes on some pages left, on some right, or even mix positions on a page. Thus, footnote handling in right-to-left context sometimes needs manual intervention. This is described in what follows.

9.1 Horizontal footnote position

When right-to-left languages are used, the `\footnote` command becomes sensitive to the text directionality. The footnote is always placed on the side that is currently the origin of direction: on the left side of the page in LTR paragraphs and on the right in RTL paragraphs.

`\RTLfootnote`
`\LTRfootnote`

For cases where this is not desired, two additional footnote commands are provided: `\RTLfootnote` and `\LTRfootnote`. `\LTRfootnote` always places the footnote on the left side, notwithstanding the current directionality. Likewise, `\RTLfootnote` always places it on the right side. Like `\footnote`, `\RTLfootnote` and `\LTRfootnote` provide an optional argument to customize the number.

9.2 Footnote rule length and position

The default placement of the footnote rule differs in XeTeX and LuaTeX output (this is due to differences in the bidi and luabidi packages). With XeTeX, footnote rules are always placed left, which is often wrong in RTL context. With LuaTeX, by contrast, the rule is placed always right if the main language is a right-to-left language, and always left if the main language is a left-to-right language, which is the right thing in many cases.

In both cases, you can change the default behavior as follows:

- `\leftfootnoterule` Put `\leftfootnoterule` in the preamble to have all rules left-aligned.
- `\rightfootnoterule` Put `\rightfootnoterule` in the preamble to have all rules right-aligned.
- `\autofootnoterule` Put `\autofootnoterule` in the preamble to have automatic placement depending on the context (see below for elaboration).
- `\textwidthfootnoterule` Put `\textwidthfootnoterule` in the preamble to have a rule that spans the whole text width.

With `\autofootnoterule`, the first footnote on the current page determines the placement. Note that this automatic can fail with footnotes at page boundaries that differ in directionality from the first footnote on the page. You can work around such cases by switching to `\rightfootnoterule` or `\leftfootnoterule` on these pages.

Note also that the rule switches might interfere in bad ways with packages or classes that redefine footnotes themselves. This is also the reason why `\autofootnoterule` is not used by default.

10 Calendars

10.1 Hebrew calendar (hebrewcal.sty)

The package `hebrewcal.sty` is almost a verbatim copy of `hebcal.sty` that comes `\Hebrewtoday` with babel. The command `\Hebrewtoday` formats the current date in the Hebrew calendar (depending of the current writing direction this will automatically set either in Hebrew script or in roman transliteration).

10.2 Islamic calendar (hijrical.sty)

This package computes dates in the lunar Islamic (Hijra) calendar.[19] It provides two macros for the end-user. The command

[19]It makes use of the arithmetical algorithm in chapter 6 of Reingold & Gershowitz, *Calendrical calculation: the Millenium edition* (Cambridge University Press, 2001).

\HijriFromGregorian `\HijriFromGregorian{⟨year⟩}{⟨month⟩}{⟨day⟩}`

\Hijritoday v1.1.1 sets the counters `Hijriday`, `Hijrimonth` and `Hijriyear`. `\Hijritoday` formats the Hijri date for the current day. This command is now locale-aware ←: its output will differ depending on the currently active language. Presently polyglossia's language definition files for Arabic, Farsi, Urdu, Turkish and Malay provide a localized version of `\Hijritoday`. If the formatting macro for the current language is undefined, the Hijri date will be formatted in Arabic or in roman transliteration, depending of the current writing direction. You can define a new format or redefine one with the command

\DefineHijriDateFormat `\DefineHijriDateFormat{⟨lang⟩}{⟨code⟩}`.

The command `\Hijritoday` also accepts an optional argument to add or subtract a correction (in days) to the date computed by the arithmetical algorithm.[20] For instance if `\Hijritoday` yields the date "7 Rajab 1429" (which is the date that was displayed on the front page of aljazeera.net on 11th July 2008), `\Hijritoday[1]` would rather print "8 Rajab 1429" (the date indicated the same day on the site gulfnews.com).

10.3 Farsi (jalālī) calendar (farsical.sty)

This package is an almost verbatim copy of `Arabiftoday.sty` (in the Arabi package), itself a slight modification of `ftoday.sty` in FarsiTEX.[21] Here we have renamed the command `\ftoday` to `\Jalalitoday`. Example: today is 18 Mihr 1399.

\Jalalitoday

11 Auxiliary commands

The macro

\charifavailable v1.47 `\charifavailable{⟨char code⟩}{⟨substitution⟩}` ←

checks whether the character with the specified `⟨char code⟩` (*i.e.*, unicode utf-16 code without preceding `0x`) exists in the current font. If so, the character is printed, if not, the `⟨substitution⟩` is printed.

Example: `\charifavailable{1E9E}{SS}` prints the capital version of the German letter ⟨ß⟩ if available (*i.e.*, ẞ), else it prints the substitution digraph SS.

[20] The Islamic calendar is indeed a purely lunar calendar based on the observation of the first visibility of the lunar crescent at the beginning of the lunar month, so there can be differences between different localities, as well as between civil and religious authorities.

[21] One day we may rewrite farsical from scratch using the algorithm in Reingold & Gershowitz (ref. n. 19).

12 Accessing language information

The following is specifically relevant to package authors who need information about the languages in use. In order to get such information, polyglossia provides the following macros:

\languagename
- `\languagename` stores the currently active (polyglossia) language name.

\mainlanguagename
- `\mainlanguagename` stores the (polyglossia) language name of the main document language.

\languagevariant
- `\languagevariant` stores the language variant if set. The macro is empty if no variant has been set.

\mainlanguagevariant
- `\mainlanguagevariant` stores the language variant of the main document language if set. The macro is empty if no variant has been set.

\babelname
- `\babelname` stores the corresponding name of the currently active language (variant) in babel. This might not only be useful if you want to support both babel and polyglossia, but also since this name is unique for a given language variety (*e.g.*, ngerman, german, swissgerman etc.). Note that this macro is also defined for languages that are not supported in babel. In that case, they are equal to the polyglossia language name.

\mainbabelname
- `\mainbabelname` analogously stores the name of document's main language (variant) in babel.

\languageid{⟨type⟩}
v1.47
- `\languageid{⟨type⟩}` ⟵ stores the identifier tag of the current language. Currently supported `⟨types⟩`:
  - `bcp-47` (alias `bcp47`): IETF BCP-47 language identifier

\mainlanguageid{⟨type⟩}
v1.47
- `\mainlanguageid{⟨type⟩}` ⟵ stores identifier tag of the main language. Currently supported `⟨types⟩`: see `\languageid`.

If you want to have a full list of loaded languages/variants, use the following macros:

\xpg@loaded
- `\xpg@loaded` stores a comma-separated list of all loaded languages (polyglossia name)

\xpg@vloaded
- `\xpg@vloaded` stores a comma-separated list of all loaded variants

\xpg@bloaded
- `\xpg@bloaded` stores a comma-separated list of babel names of all language variants

\xpg@bcp@loaded
v1.47
- `\xpg@bcp@loaded` ⟵ stores a comma-separated list of the BCP-47 IDs of all language variants

Whether a language is loaded can be tested by

\iflanguageloaded

```
\iflanguageloaded{⟨lang⟩}{⟨true⟩}{⟨false⟩}
```

where `⟨lang⟩` is a polyglossia language name, by

`\ifbabellanguageloaded` `\ifbabellanguageloaded{⟨lang⟩}{⟨true⟩}{⟨false⟩}`

where ⟨lang⟩ is a babel language name (see table 2 on p. 5), or by

`\iflanguageidloaded` v1.47 `\iflanguageidloaded{⟨type⟩}{⟨id⟩}{⟨true⟩}{⟨false⟩}` ←

where ⟨type⟩ is a supported language id type (such as `bcp-47`) and ⟨id⟩ is a language id (such as `en-US`; see table 3 on p. 6).

Finally, if you want to know whether a specific language option has been set, you can use

`\iflanguageoption` v1.47 `\iflanguageoption{⟨lang⟩}{⟨opt. key⟩}{⟨opt. value⟩}{⟨true⟩}{⟨false⟩}` ←

13 Acknowledgements (by François Charette)

Polyglossia is notable for being a recycle box of previous contributions by other people. I take this opportunity to thank the following individuals, whose splendid work has made my task almost trivial in comparision: Johannes Braams and the numerous contributors to the babel package (in particular Boris Lavva and others for its Hebrew support), Alexej Kryukov (antomega), Will Robertson (fontspec), Apostolos Syropoulos (xgreek), Youssef Jabri (arabi), and Vafa Khalighi (xepersian and bidi). The work of Mojca Miklavec and Arthur Reutenauer on hyphenation patterns with their package hyph-utf8 is of course invaluable. I should also thank other individuals for their assistance in supporting specific languages: Yves Codet (Sanskrit), Zdeněk Wagner (Hindi), Mikhal Oren (Hebrew), Sergey Astanin (Russian), Khaled Hosny (Arabic), Sertaç Ö. Yildiz (Turkish), Kamal Abdali (Urdu), and several other members of the XeTeX user community, notably Enrico Gregorio, who has sent me many useful suggestions and corrections and contributed the `\newXeTeXintercharclass` mechanism in xelatex.ini which is now used by polyglossia. More recently, Kevin Godby of the Ubuntu Manual project has contributed very useful feedback, bug hunting and, with the help of translators, new language definition files for Asturian, Lithuanian, Occitan, Bengali, Malayalam, Marathi, Tamil, and Telugu. It is particularly heartening to realize that this package is used to typeset a widely-read document in dozens of different languages! Support for Lao was also added thanks to Brian Wilson. I also thank Alan Munn for kindly proof-reading the penultimate version of this documentation. And of course my gratitude also goes to Jonathan Kew, the formidable author of XeTeX!

14 More acknowledgements (by the current development team)

Many thanks to all the people who have contributed bugfixes and new features to polyglossia since we took over. In alphabetical order: Ignas Anikevicius, Sina Ahmadi, Wouter Bolsterlee, Christian Buhtz, Zgarbul Andrey, Oleg Domanov, Philipp Gesang, Kevin Godby, Enrico Gregorio, Khaled Hosny, Najib Idrissi, user julroy67, Dohyun Kim, Phil Kime, Mike Kroutikov, Ivan Kokan, Caleb Maclennan, José Mancera, Miquel Ortega, Yevgen Pogribnyi, Will Robertson, Maïeul Rouquette, Elie Roux, Hugo Roy, Guy Rutenberg, Philipp Stephani, Niranjan Tambe, Keno Wehr, Dominik Wujastyk, Sertaç Ö. Yildiz, Maksim Zholudev, Yan Zhou, and Stefan Zlatinov. Their respective contributions can be identified from the contributor statistics on GitHub.

Among the ones who sent contributions directly to us we would like to especially thank Claudio Beccari, the indefatigable champion of Romance languages, and beyond! Furthermore, kudos go to Moritz Wemheuer (of biblatex) who has helped a lot to improve polyglossia interaction with biblatex and csquotes.

Not at least, we are very grateful for all bug reports and feature enhancement requests we received from the numerous users we cannot list all here (but again, you can find all names on GitHub). Please go on with that, you are keeping polyglossia running!