A Babel language definition file for French

frenchb.dtx v4.1a, 2026-06-06

Daniel Flipo
daniel.flipo@free.fr

Contents

1 The French language

The file frenchb.dtx[1], defines all the language definition macros for the French language.
Customisation for the French language is achieved following the book "Lexique des règles typographiques en usage à l'Imprimerie Nationale'' troisième édition (1994), ISBN-2-11-081075-0.
First version released: 1.1 (May 1996) as part of Babel-3.6beta. Version 2.0a was released in February 2007 and version 3.0a in February 2014.

This file frenchb.dtx is for LuaTeX,
See file frenchb3.dtx
for pdfTeX and XeTeX.

Significant changes, listed in section 1.4, p. 12, have occurred in version 4.1a, please read section 1.4 if your documents have French as main language.

babel-french has been improved using helpful suggestions from many people, mainly from Jacques André, Michel Bovani, Thierry Bouche, Vincent Jalby, Denis Bitouzé, Thomas Savary, Ulrike Fisher and Marcel Krüger. Thanks to all of them!
An extensive documentation in French (file frenchb-doc.pdf) is now included in babel-french.

1.1 Basic interface

In a multilingual document, some typographic rules are language dependent, i.e. spaces before 'high punctuation' (: ; ! ?) in French, others modify the general layout (i.e. layout of lists, footnotes, indentation of first paragraphs of sections) and should apply to the whole document.
The French language can be loaded with Babel by a command like:
`\usepackage[german,spanish,french,british]{babel}` [2]
babel-french takes account of Babel's *main language* defined as the *last* option at Babel's loading or set through the `\DocumentMetadata{lang=..}` command. When French is not Babel's main language, babel-french does not alter the general layout of the document (even in parts where French is the current language): the layout of lists, footnotes, indentation of first paragraphs of sections are not customised by babel-french.

[1] The file described in this section has version number v4.1a and was last revised on 2026-06-06.
[2] *Always* use french as option name for the French language, former aliases frenchb, francais and acadian have finally been removed as announced several years ago!

When French is Babel's main language, babel-french makes the following changes to the global layout, *both in French and in all other languages*[3]:

1. the first paragraph of each section is indented (LaTeX only);
2. the default items in itemize environment are set to '—' instead of '•', and all vertical spacing and glue is deleted; it is possible to change '—' to something else ('–' for instance) using \frenchsetup{} (see section 1.2 p. 5);
3. vertical spacing in general LaTeX lists is shortened;
4. footnotes are displayed "à la française".

Regarding local typography, the command \selectlanguage{french} switches to the French language[4], with the following effects:

1. French hyphenation patterns are made active;
2. 'high punctuation' characters (: ; ! ?) automatically add correct spacing in French; this is achieved using callbacks in Lua(La)TeX, these characters are no longer made "active";
3. \today prints the date in French;
4. the caption names are translated into French (LaTeX only). For customisation of caption names see section 1.2.2 p. 9.
5. the space after \dots is removed in French.

Some commands are provided by babel-french to make typesetting easier:

1. French quotation marks can be entered using the command \frquote{}: \frquote{some text} will output « some text ». Former commands \og and \fg are kept for backward compatibility only: \og some text\fg{} produces the same output as \frquote{some text}.

   If French quote characters are available on your keyboard, you can use them, the required nobreak spaces will be added automatically: you can type either « guillemets » or «guillemets» [5] (with or without spaces) to get properly typeset French quotes. The same is true for the single guillemets ‹ and ›.

   For quotations spreading over more than one paragraph, \frquote will add at the beginning of every paragraph of the quotation either an opening French guillemet («), or a closing one (») or nothing depending on option EveryParGuill=open or =close or =none, see p. 8.
   The command \NoEveryParQuote is provided to locally suppress unwanted guillemets (typically when lists are embedded in \frquote{}), it must be used inside an environment or a group *only*.

[3] For each item, hooks are provided to reset standard LaTeX settings or to emulate the behaviour of former versions of babel-french (see command \frenchsetup{}, section 1.2 p. 5).

[4] \selectlanguage{francais} and \selectlanguage{frenchb} are no longer supported.

[5] Or even «~guillemets~»…

`\frquote` is recommended to enter embedded quotations “à la française”, several variants are provided through options:

- the inner quotation is surrounded by double quotes (“*texte*”) unless option `InnerGuillSingle=true`, then a) the inner quotation is printed as ‹*texte*› and b) if the inner quotation spreads over more than one paragraph, every paragraph included in the inner quotation starts with a ‹ or a › or nothing, depending on option `EveryParGuill=open` (default) or `=close` or `=none`.
- it is possible to add a French opening or closing guillemet (« or ») at the beginning of every line of the inner quotation using option `EveryLineGuill=open` or `=close`; note that with any of these options, the inner quotation is surrounded by French guillemets (« and ») regardless option `InnerGuillSingle`; the default is `EveryLineGuill=none`.

A starred variant `\frquote*` is meant for inner quotations which end together with the outer one: using `\frquote*` for the inner quotation will print only one closing quote character (the outer one) as recommended by the French ‘Imprimerie Nationale’.

2. `\frenchdate{`*<year>*`}{`*<month>*`}{`*<day>*`}` helps typesetting dates in French: `\frenchdate{2001}{01}{01}` will print 1er janvier 2001 in a box without any linebreak.

3. A command `\up` is provided to typeset superscripts like `M\up{me}` (abbreviation for “Madame”), `1\up{er}` (for “premier”). Other commands are also provided for ordinals: `\ier`, `\iere`, `\iers`, `\ieres`, `\ieme`, `\iemes` (`3\iemes` prints 3es). All these commands take advantage of real superscript letters when they are available in the current font.

4. Command `\bname{}` (boxed name) is provided to typeset family names: its argument will not be hyphenated except on explicit hyphens. `\bsc{}` (boxed small caps) is a variant that prints its argument in small capitals, it is meant for bibliographies, signatures, etc. Usage: `Albert~\bsc{Camus}`.

5. Commands `\primo`, `\secundo`, `\tertio` and `\quarto` print 1o, 2o, 3o, 4o. `\FrenchEnumerate{6}` prints 6o.

6. Abbreviations for “Numéro(s)” and “numéro(s)” (No Nos no and nos) are obtained via the commands `\No`, `\Nos`, `\no`, `\nos`.

7. Two commands `\degre` and `\degres` are provided (for backward compatibility only) to typeset the symbol “degré”. Entering the raw character ° is easier.

8. In math mode the comma has to be surrounded with braces to avoid a spurious space being inserted after it, in decimal numbers for instance (see the TEXbook p. 134). The command `\DecimalMathComma` makes the comma behave as an ordinary character *when the current language is French* (no space added); as

a counterpart, if \DecimalMathComma is active, an explicit thin space has to be added in lists and intervals: $(x,\,y)$, $[0,\,1]$. \StandardMathComma switches back to the standard behaviour of the comma in French.

The icomma package is an alternative workaround.

9. A command \nombre was provided in 1.x versions to easily format numbers in slices of three digits separated either by a comma in English or with a space in French; \nombre is now mapped to \numprint from numprint.sty, which should be loaded *after* Babel, see numprint.pdf for more information.

10. babel-french has been designed to take advantage of the xspace package if present: adding \usepackage{xspace} in the preamble will force macros like \fg, \ier, \ieme, \dots, …, to respect the spaces you type after them, for instance typing '1\ier juin' will print '1er juin' (no need for a forced space after 1\ier).

1.2 Customisation

Customisation of babel-french relies on command \frenchsetup{} (formerly called \frenchbsetup{}, the latter name will be kept for ever to ensure backwards compatibility), options are entered using the l3keys syntax. The command \frenchsetup{} is to appear in the preamble only (after loading Babel).

1.2.1 \frenchsetup{options}

\frenchbsetup{} and \frenchsetup{} are synonymous; the latter should be preferred as the language name for French in Babel is no longer frenchb but french. \frenchsetup{ShowOptions} prints all available options to the .log file, it is just meant as a remainder of the list of offered options. As usual with l3keys syntax, boolean options (as ShowOptions) can be entered as ShowOptions=true or just ShowOptions, the =true part can be omitted.

The other options are listed below. Their default value is shown between braces, sometimes followed be a '*'. The '*' means that the default shown applies when babel-french is loaded as the *last* option of Babel —Babel's *main language*—, and is toggled otherwise.

General Layout

StandardLayout=true (false*) forces babel-french not to interfere with the layout: no action on any kind of lists, first paragraphs of sections are not indented (as in English), no action on footnotes; it useless unless French is the main language. This option can be used to avoid conflicts with classes or packages which customise lists or footnotes.

`IndentFirst=false (true*)`; set this option to `false` if you do not want `babel-french` to force indentation of the first paragraph of sections. When French is the main language, this option applies to all languages.

`PartNameFull=false (true)`; when true (the default), `babel-french` numbers the title of `\part{}` commands as "Première partie", "Deuxième partie" and so on. With some classes which change the `\part{}` command (AMS classes do so), you could get "Première partie 1", "Deuxième partie 2" in the toc; when this occurs, this option should be set to `false`, part titles will then be printed as "Partie I", "Partie II".

`TocPartNameFull=false (true*)`; when true (default), parts are also numbered "Première partie", "Deuxième partie", in the table of contents. This works currently only for the `memoir` and koma-script classes (standard classes do not provide (yet?) any hook to customise the TOC). `babel-french` provides a command `\FBtocpartname{`*<Romannum>*`}` which returns the formatted string ("Deuxième partie" if the argument is "II"), it can be used with all classes; it is possible to add something (colon, dot,…) at the end of the string by redefining `\FBtocpartsep`: `\renewcommand*{\FBtocpartsep}{. }` adds a dot.

Lists Layout [6]

`ListItemsAsPar=true (false)`; setting this option to `true` is recommended: list items will be displayed as paragraphs with indented labels (in the "Imprimerie Nationale" way) instead of having labels hanging into the left margin. How these two layouts differ is shown below:

Text starting at 'parindent'
<= Leftmargin
— first item running on two lines or more…
— first second level item on two lines…
— next one…
— second item…

Default French layout

Text starting at 'parindent'
<= Leftmargin
— first item running on two lines or more…
— first second level item on two lines…
— next one…
— second item…

With `ListItemsAsPar=true`

`StandardListSpacing=true (false*)`[7]; `babel-french` usually customises the vertical spaces in the `list` environment, this affects all lists, including `itemize`, `enumerate`, `description`, but also `abstract`, `quote`, `quotation`, `verse`, etc. which are based on `list`. Setting this option to `true` reverts to the standard settings of the `list` environment as defined by the document class.

[6] Two options, NosepItemize and NosepEnumerate, require the new templates, *i.e.* a `\DocumentMetadata{}` command.

[7] This option should be used instead of former option `ReduceListSpacing` (kept for backward compatibility) which could be misleading: with some classes (smfart, smfbook f.i.) you had to set `ReduceListSpacing=false` to revert to the class settings which actually reduce list's spacings even more than `babel-french`! `StandardListSpacing=true` replaces `ReduceListSpacing=false`.

`NosepItemize=true (true*)`; turning this option to `false` slightly enlarges the vertical spaces included in `itemize` lists to match their counterparts in `enumerate` and `description` lists.

`NosepEnumerate=true (false*)`; turning this option to `true` suppresses any vertical space in `enumerate` and `description` lists, to match `itemize` lists.

`StandardItemLabels=true (false*)`; when set to `true` this option prevents `babel-french` from changing the labels in `itemize` lists in French.

`ItemLabels=\textbullet, \textendash, \ding{43}, (\textemdash*)`; when `StandardItemLabels=false` (the default), this option enables to choose the label used in French `itemize` lists for all levels. The next four options do the same but each one for a specific level only. Note that `\ding{43}` requires loading the `pifont` package.

`ItemLabeli=\textbullet, \textendash, \ding{43} (\textemdash*)`

`ItemLabelii=\textbullet, \textendash, \ding{43} (\textemdash*)`

`ItemLabeliii=\textbullet, \textendash, \ding{43} (\textemdash*)`

`ItemLabeliv=\textbullet, \textendash, \ding{43} (\textemdash*)`

`StandardLists=true (false*)` forbids `babel-french` to customise any kind of list. The option `StandardLists=true` should be used in case of conflicts with classes or packages that customise lists too.

Footnotes layout

`FrenchFootnotes=false (true*)` reverts to the standard layout of footnotes. By default `babel-french` typesets leading numbers as '`1. `' instead of '1', but does not change footnotes numbered with symbols (as in the `\thanks` command) or with letters (as in minipages).

`AutoSpaceFootnotes=false (true*)`; by default `babel-french` adds a (customisable) thin space in the running text before the number or symbol calling the footnote. Making this option `false` reverts to the standard setting (no space added). The default definition of this thin space is:
`\newcommand*{\FBfnmarkspace}{\kern .5\fontdimen2\font}`

Punctuation

`AutoSpacePunctuation=false (true)`; with Lua(La)TeX changing this option to `false` doesn't make sense as the LuaTeX callback takes care of special cases where no space should be added: URLs (http://mysite), in MS-DOS paths (C:\Foo) or in timetables (10:55). .

`ThinColonSpace=true (false)` changes the non-breaking space added before the colon ':' to a thin space, so that the same amount of space is added before any of the four 'high punctuation' characters. The default setting is supported by the French 'Imprimerie Nationale'.

`OriginalTypewriter=true (false)` prevents any customisation of `\ttfamily` and `\texttt{}` in French. This option should only be used to ensure backward compatibility. The current default behaviour is to switch off any addition of space before high punctuation with typewriter fonts (*e.g.* verbatim).

`UnicodeNoBreakSpaces=true (false)`; (experimental) this option should be set to `true` *only while converting LuaLaTeX files* to HTML. It ensures that non-breaking spaces added by `babel-french` are inserted in the PDF file as U+A0 or U+202F (thin) instead of penalties and glues. Note that `lwarp` (v. 0.37 and up) is fully compatible with `babel-french` for translating PDFLaTeX or XeLaTeX files to HTML.

French quotes

`og=«, fg=»`; this option has been kept for backward compatibility but has no effect in Lua(La)TeX, it just prints a warning in the `.log` file.

`INGuillSpace=true (false)` resets the dimensions of spaces after opening French quotes and before closing French quotes to the French 'Imprimerie Nationale' standards (inter-word space). `babel-french`'s default setting produces slightly narrower spaces with less stretchability.

`EveryParGuill=open, close, none (open)`; sets whether an opening quote («) or a closing one (») or nothing should be printed by `\frquote{}` at the beginning of every paragraph included in a level 1 (outer) quotation. This option is also considered for level 2 (inner) quotations to decide between ‹ and › when `InnerGuillSingle=true` (see below).

`EveryLineGuill=open, close, none (none)`; with LuaTeX based engines *only*, it is possible to set this option to `open` [resp. `close`]; this ensures that a '«' [resp. '»'] followed by a proper space will be inserted at the beginning of every line of embedded (inner) quotations spreading over more than one line (provided that both outer and inner quotations are entered with `\frquote{}`). When `EveryLineGuill=open` or `=close` the inner quotation is always surrounded by « and », the next option is ineffective.

`InnerGuillSingle=true (false)`; if `InnerGuillSingle=false` (the default), inner quotations entered with `\frquote{}` start with ``` `` ``` and end with `''`. If `InnerGuillSingle=true`, ‹ and › are used instead of British double quotes; moreover if option `EveryParGuill=open` (or `close`) is set, a ‹ (or ›) is added at the beginning of every paragraph included in the inner quotation.

Numbers

`ThinSpaceInFrenchNumbers=true (false)`; if `numprint` has been loaded with the `autolanguage` option, while typesetting numbers with the `\numprint{}` command, `\npthousandsep` is defined as a non-breaking space (~) [8] in French;

[8] Actually without stretch nor shrink.

when set to true, this option redefines \npthousandsep as a thin space (\FBthinspace).

Figure and table captions

SmallCapsFigTabCaptions=false (true*); when set to false, \figurename and \tablename will be printed in French captions as "Figure" and "Table" instead of being printed in small caps (the default). The same result can be achieved by defining \FBfigtabshape as \relax before loading babel-french (in a document class f.i.).

CustomiseFigTabCaptions=true (false); this option is now false, as the colon in captions is no printed properly in French with LuaTeX; turning it to true prints a warning and changes the caption's separator into endash to mimic former versions of babel-french. Not recommended, see below (section 1.2.3) for hints to customise captions.

Superscripts

FrenchSuperscripts=false (true); the babel-french' \up{} command should print better superscripts than \textsuperscript; turning this option to false redefines \up{} as \textsuperscript (not recommended, except if \up{} fails).

LowercaseSuperscripts=false (true); by default babel-french inhibits the uppercasing of superscripts (for instance when they are moved to page headers). Making this option false will disable this behaviour (not recommended).

Warnings

SuppressWarning=true (false); can be turned to true if you are bored with babel-french's warnings; use this option as *first* option of \frenchsetup{} to cancel warnings launched by other options.

Options' order — Please remember that options are read in the order they appear in the \frenchsetup{} command. Someone wishing that babel-french leaves the layout of lists and footnotes untouched but caring for indentation of first paragraph of sections should choose \frenchsetup{StandardLayout,IndentFirst}.
The reverse order \frenchsetup{IndentFirst,StandardLayout} would lead to option IndentFirst being overwritten by StandardLayout.

1.2.2 Caption names

All caption names can easily be customised in French using the simplified syntax introduced by Babel 3.9, for instance \def\frenchproofname{Preuve}. The older syntax \addto\captionsfrench{\def\proofname{Preuve}} still works.

1.2.3 Figure and table captions

Most document classes use a colon as captions' separator in figures and tables like this: 'Figure 1: '. With 8-bits engines (TeX, pdfTeX) the colon was made active too late to ensure a proper space before it. The problem has vanished with LuaTeX. Therefore, the former patches provided in the legacy versions of babel-french have been dropped: \CaptionSeparator is no longer defined and the CustomiseFigTabCaptions option is now turned to false by default. Switching it to true, prints a Warning in the .log file and currently turns the captions' separator into an endash (this might change in the future).
Customisation of the captions' separator should be achieved outside babel-french; here are some hints for those who want to get the endash formerly provided by babel-french:

- with standard classes article, book, report, use caption.sty:
  \usepackage[labelsep=endash]{caption}
- with the memoir class, just add:
  \captiondelim{\space\textendash\space}
- with the koma-script classes, just add:
  \renewcommand{\captionformat}{\space\textendash\space}
- with the beamer class, just add:
  \setbeamertemplate{caption label separator}[endash]

Following the IN's recommendations, \figurename and \tablename should be typeset in small caps in French, babel-french provides the SmallCapsFigTabCaptions option (default is true) to do so. It can be set to false to typeset \figurename and \tablename in French as "Figure'' and "Table'' rather than in small caps (the default).

1.3 Hyphenation checks

Once you have built your format, a good precaution would be to perform some basic tests about hyphenation in French. For LaTeX2e I suggest this:

- run LuaLaTeX on the following file:

```
%%% Test file for French hyphenation.
\documentclass[french]{article}
\usepackage{fontspec}       % mandatory for French
\setmainfont{NewCM10-Book}  % or erewhon, XCharter…
\usepackage{babel}
\begin{document}
\showhyphens{signal container événement algèbre}
\end{document}
```

- check the hyphenations proposed by TEX in your log-file; in French you should get
  `si-gnal contai-ner évé-ne-ment al-gèbre`.
  Do not care about how accented characters are displayed in the log-file, what matters is the position of the '-' hyphen signs *only*.

If they are all correct, your installation (probably) works fine, if one (or more) is (are) wrong, ask a local wizard to see what's going wrong and perform the test again (or e-mail me about what happens).
Possible mismatches: you get `sig-nal con-tainer`, this probably means that the hyphenation patterns you are using are for US-English, not for French.

1.4 Changes

What's new in version 4.1?

Options acadian, canadian, frenchb and francais are no longer supported, the only possible Babel's option for French is french; anyway babel-french is customisable enough to fit any French dialect, see \fbsetup{} p.42.
If French is *not* the document's main language, the layout remains unchanged, *i.e.* is unaffected by the changes introduced in this version.
This version aims at compatibility with the new templates introduced by the LaTeX Team for the PDF tagging project. These new templates are part of the 2026-06-01 LaTeX release.
As LuaTeX users may add a \DocumentMetadata{ ... } command (involving the new templates) or not (then legacy code is used), new code has been added to enable the customisation of lists and footnotes using the new templates.
When a \DocumentMetadata{ ... } command is detected, french.ldf automatically switches from legacy to new code. A warning is issued in case the LaTeX format is older than 2026-06-01.
The new templates have positive side-effects, babel-french becomes f.i. compatible with enumitem and footmisc.

What's new in version 4.0?

babel-french has been split into two files frenchb3.dtx, the legacy part, which is frozen, is meant for TeX, pdfTeX and XeTeX engines, and frenchb.dtx for LuaTeX *only*.
This has made possible to deeply simplify the current file frenchb.dtx, stripping old code which no longer makes sense with Lua(La)TeX.
Consequently, some \frenchsetup{} options have been modified, deleted or added:

- AutoPunctuation *should not be turned to* false with LuaLaTeX, a warning in .log file is issued if you do so; frenchb.lua now handles automatically the special cases (2:1, \http://, C:\, !!, etc.) requiring no space before high punctuation.
- \frenchsetup{og=«, fg=»} is useless, it just prints a warning; single and double French quotes (« and », ‹ and ›) automatically add the required spaces, it is still possible to inhibit this locally using {\NoAutospacing }.
- CustomiseFigTabCaptions is now false, it means that babel-french no longer customises the captions' separator (usually a colon); when forced to true, it issues a warning and turns the separator to en endash, see section 1.2.3 for better options.
- Options OldFigTabCaptions, ListOldLayout and GlobalLayoutFrench have been deleted (they emulated very old behaviours of babel-french).

- a new option TocPartFullName has been added to enhance PartFullName. When true (the default), the numbered parts are printed as “Première partie”, “Deuxième partie” in the table of contents too. This works currently only with the memoir and koma-script classes.

frenchb.lua has a new function euphonic_t to deal with compound words’ hyphenation like “va-t-on”, “semble-t-il” etc. A bug occuring in case \spaceskip is not null has been fixed.
Version 4.0b takes advantage of the new footnotes’ template (when it is available) to customise the footnotes’ layout. This should fix issue 932.
Version 4.0: commands \StandardFootnotes and \FrenchFootnotes are no longer available when neww templates are used: footnotes inside minipages are now numbered the standard way with both new and old interface.

What’s new in version 3.7?

The acadian dialect is no longer supported: \usepackage[acadian]{babel} prints a warning and uses french instead. Reason: I have never got feedback from anybody using them; anyway babel-french is customisable enough to fit any French dialect, see \fbsetup{} p.42.
Version 3.7 is the frozen version in frenchb3.dtx.

What’s new in version 3.6?

Version 3.6a no longer loads the keyval package, replaced by core LaTeX commands (l3keys). The thin space added before footnote’s calls is now customisable (suggested by Thomas Savary), the command’s name is \FBfnmarkspace.

2 The code

2.1 Initial setup

The macro \LdfInit takes care of preventing that this file is loaded more than once, checking the category code of the @ sign, etc.

```
<*french>
\LdfInit\CurrentOption{FBclean@on@exit}
```

If the engine is not LuaTeX, revert to the version 3.7 of babel-french.

```
\let\bbl@tempa\relax
\ifnum\bbl@engine=\@ne \else \input french3.ldf\relax
  \let\bbl@tempa\endinput
\fi
\bbl@tempa
```

The rest of this file is *only for the Luatex engine*.
Let's provide a substitute for \PackageError, \PackageWarning and \PackageInfo not defined in Plain:

```
\def\fb@error#1#2{%
    \begingroup
      \newlinechar=`\^^J
      \def\\{^^J(french.ldf) }%
      \errhelp{#2}\errmessage{\\#1^^J}%
    \endgroup}
\def\fb@warning#1{%
   \begingroup
      \newlinechar=`\^^J
      \def\\{^^J(french.ldf) }%
      \message{\\#1^^J}%
   \endgroup}
\def\fb@info#1{%
   \begingroup
      \newlinechar=`\^^J
      \def\\{^^J}%
      \wlog{#1}%
   \endgroup}
```

Quit if the LuaTeX engine is too old.

```
\ifnum\luatexversion<100
    \ifx\PackageWarning\@undefined
      \fb@warning{Please upgrade LuaTeX to version 1.1 or above!\\%
         Aborting.}%
    \else
      \PackageWarning{french.ldf}{Please upgrade LuaTeX
         to version 1.1 or above!\MessageBreak
```

```
        Aborting. Reported}%
    \fi
  \let\bbl@tempa\endinput
\fi
\bbl@tempa
```

Quit if Babel's version is less than 24.1.

```
\let\bbl@tempa\relax
\ifdefined\babeltags
\else
   \let\bbl@tempa\endinput
   \ifdefined\PackageError
      \PackageError{french.ldf}
         {babel-french requires babel v.24.1.\MessageBreak
          Aborting here}
         {Please upgrade Babel!}
   \else
      \fb@error{babel-french requires babel v.24.1.\\
                Aborting here}
               {Please upgrade Babel!}
   \fi
\fi
\bbl@tempa
```

Make sure that `\l@french` is defined (fallbacks are `\l@nohyphenation` if available or 0). `babel.def` (3.9i and up) defines `\l@<languagename>` also for eTeX, LuaTeX and XeTeX formats which set `\lang@<languagename>`.

```
\def\FB@nopatterns{%
   \ifdefined\l@nohyphenation
      \adddialect\l@french\l@nohyphenation
      \edef\bbl@nulllanguage{\string\language=nohyphenation}%
   \else
      \edef\bbl@nulllanguage{\string\language=0}%
      \adddialect\l@french0
   \fi
   \@nopatterns{French}}
\ifdefined\l@french \else \FB@nopatterns \fi
```

French uses the standard values of `\lefthyphenmin` (2) and `\righthyphenmin` (3); let's provide their values though, as required by Babel.

```
\providehyphenmins{french}{\tw@\thr@@}
```

`\ifLaTeXe` No support is provided for late LaTeX-2.09: issue a warning and exit if LaTeX-2.09 is in use. Plain is still supported.

```
\newif\ifLaTeXe
\let\bbl@tempa\relax
```

```
\ifdefined\magnification
\else
  \ifdefined\@compatibilitytrue
    \LaTeXetrue
  \else
    \PackageError{french.ldf}
       {LaTeX-2.09 format is no longer supported.\MessageBreak
        Aborting here}
       {Please upgrade to LaTeX2e!}
    \let\bbl@tempa\endinput
  \fi
\fi
\bbl@tempa
```

\ifFBfrench True when the current language is French; will be set to true by \extrasfrench and to false by \noextrasfrench. Used in \DecimalMathComma.

```
\newif\ifFBfrench
```

\extrasfrench \noextrasfrench The macro \extrasfrench will perform all the extra definitions needed for the French language. The macro \noextrasfrench is used to cancel the actions of \extrasfrench.

In French, character “apostrophe’’ (U+27 or U+2019) is a letter in expressions like l’ambulance (French hyphenation patterns provide entries for this kind of words). This means that the \lccode of “apostrophe’’ has to be non null in French for proper hyphenation of those expressions, and has to be reset to null when exiting French.

```
\def\extrasfrench{%
    \FBfrenchtrue
    \babel@savevariable{\lccode"27}%
    \lccode"27="27
    \babel@savevariable{\lccode"2019}%
    \lccode"2019="2019
    \bbl@frenchspacing
}
\def\noextrasfrench{\FBfrenchfalse \bbl@nonfrenchspacing}
```

2.2 Punctuation

With LuaTeX, callbacks are used to get rid of active punctuation.

\FBguillspace \FBcolonspace \FBthinspace In French high punctuation characters (: ; ! ?) and guillemets require some space to be added before them (: ; ! ? » ›) or after them («, ‹). Following the I.N. specifications, the ‘:’ requires an inter-word space before it, the other three require just a thin space. So we define \FBcolonspace as \space (inter-word space) and \FBthinspace as an half inter-word space with no shrink nor stretch. \FBguillspace is meant for guillemets; it has been fine tuned by Thierry Bouche to 80% of an inter-word space with

reduced stretchability. All three are user customisable in the preamble, best using the \FBsetspaces command described below. These three commands are designed for basic French. Other French dialects can use different settings, see below.
A penalty will be added before these spaces to prevent line breaking.

```
\newcommand*{\FBguillspace}{\hskip .8\fontdimen2\font
                            plus .3\fontdimen3\font
                           minus .8\fontdimen4\font \relax}
\newcommand*{\FBcolonspace}{\space}
\newcommand*{\FBthinspace}{\hskip .5\fontdimen2\font \relax}
```

\FBsetspaces This command makes it easy to fine tune \FBguillspace, \FBcolonspace and \FBthinspace in French using the optional argument. It is meant for LaTeX2e *only* and can only be used in the preamble. Four mandatory arguments[9]: the first one is a *string* either "guill", "colon", or "thin", the last three are decimal numbers specifying *width*, *stretch* and *shrink* relative to the relevant *fontdimens*. For instance \FBsetspaces{colon}{0.5}{0}{0} defines \FBcolonspace as a thinspace as suggested by the "Guide du typographe Roman".

```
\ifLaTeXe
  \newcommand*{\FBsetspaces}[5][french]{%
    \@namedef{FB#2space}{\hskip #3\fontdimen2\font
                         plus #4\fontdimen3\font
                        minus #5\fontdimen4\font \relax}}
  \@onlypreamble\FBsetspaces
\fi
```

We must set the LuaTeX tables for French after possible changes made in the preamble (\frenchsetup{} or \FBsetspaces{}) and before Babel switches to French at \begin{document}.

```
\ifLaTeXe
  \AddToHookNext{env/document/before}{%
    \set@glue@table{colon}%
    \set@glue@table{thin}%
    \set@glue@table{guill}%
  }
\fi
```

This code is for Plain: load ltluatex.tex if it hasn't been loaded before Babel.

```
\ifdefined\newluafunction\else
    \input ltluatex.tex
    \ifdefined\bbl@attr@locale \else \newattribute\bbl@attr@locale \fi
    \newattribute\bbl@attr@locale
\fi
```

We define five LuaTeX attributes to control spacing in French for 'high punctuation' and quotes, making sure that \newattribute is defined.

[9] The former optional lang argument no longer has any effect.

\FB@spacing=0 switches off any space tuning both before high punctuation characters and inside French quotes (i.e. function french_punctuation doesn't alter the node list at all).
\FB@addDPspace=0 switches off automatic insertion of spaces before high punctuation characters (but typed spaces are still turned into non-breaking thin- or word-spaces).
\FB@ucsNBSP triggers the replacement of glues by characters, it is controlled by option UnicodeNoBreakSpaces.

```
\newattribute\FB@spacing      \FB@spacing=\@ne
\newattribute\FB@addDPspace   \FB@addDPspace=\@ne
\newattribute\FB@ucsNBSP      \FB@ucsNBSP=\z@
```

The next command will be used in the first call of \extrasfrench to convert \FBcolonspace, \FBthinspace and \FBguillspace into a table usable by LuaTeX. This way, any customisation done in the preamble (by \frenchsetup{}, redefinitions or \FBsetspaces commands) are taken into account.
In case parsing by the Lua function FBget_glue (defined in file frenchb.lua) fails due to unexpected syntax in \FB... space the table remains unchanged and a warning is issued. The matching space characters for option UnicodeNoBreakSpaces are set as word space, thin space or null space according to the *width* parameter.

```
\newcommand*{\set@glue@table}[1]{%
  \directlua {
    local s = token.get_meaning("FB#1space")
    local t = FBget_glue(s)
    if t then
       FBsp.#1.gl = t
       if FBsp.#1.gl[1] > 0.6 then
          FBsp.#1.ch = 0xA0
       elseif FBsp.#1.gl[1] > 0.2 then
          FBsp.#1.ch = 0x202F
       else
          FBsp.#1.ch = 0x200B
       end
    else
       texio.write_nl('term and log', '')
       texio.write_nl('term and log',
         '*** french.ldf warning: Unexpected syntax in FB#1space,')
       texio.write_nl('term and log',
         '*** french.ldf warning: LuaTeX table FBsp unchanged.')
       texio.write_nl('term and log',
         '*** french.ldf warning: Consider using FBsetspaces to ')
       texio.write('term and log', 'customise FB#1space.')
       texio.write_nl('term and log', '')
    end
  }%
}
```

```
</french>
```

frenchb.lua (*env.*) This is `frenchb.lua`. It holds Lua code to deal with ‘high punctuation’ and quotes. This code is based on suggestions from Paul Isambert.

First we define two flags to control spacing before French ‘high punctuation’ (thin space or inter-word space).

```
<*lua>
local FB_punct_thin =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte(";")] = true}
local FB_punct_thick =
  {[string.byte(":")] = true}
```

Managing spacing for ‘»’ and ‘›’ (U+203A) can be done by the way; we define two flags, `FB_punct_left` for characters requiring some space before them and `FB_punct_right` for ‘«’ and ‘‹’ which must be followed by some space. In case LuaTeX is used to output T1-encoded fonts instead of OpenType fonts, codes `0x13` and `0x14` have to be added for ‘«’ and ‘»’.

```
local FB_punct_left =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte(";")] = true,
   [string.byte(":")] = true,
   [0x14]             = true,
   [0xBB]             = true,
   [0x203A]           = true}
local FB_punct_right =
  {[0x13]             = true,
   [0xAB]             = true,
   [0x2039]           = true}
```

Two more flags will be needed to avoid spurious spaces in strings like !! ?? or (?)

```
local FB_punct_null =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte("[")] = true,
   [string.byte("(")] = true,
```

or if the user has typed a non-breaking space U+00A0 or U+202F (thin) before a ‘high punctuation’ character: no space should be added by `babel-french`. Same is true inside French quotes.

```
   [0xA0]             = true,
   [0x202F]           = true}
```

```
local FB_guil_null =
  {[0xA0]             = true,
   [0x202F]           = true}
```

Local definitions for nodes:

```
local new_node     = node.new
local copy_node    = node.copy
local node_id      = node.id
local HLIST        = node_id("hlist")
local TEMP         = node_id("temp")
local DISC         = node_id("disc")
local KERN         = node_id("kern")
local GLUE         = node_id("glue")
local GLYPH        = node_id("glyph")
local PENALTY      = node_id("penalty")
local nobreak      = new_node(PENALTY)
nobreak.penalty    = 10000
local nbspace      = new_node(GLYPH)
local insert_node_before = node.insert_before
local insert_node_after  = node.insert_after
local remove_node        = node.remove
```

Commands \FBthinspace, \FBcolonspace and \FBguillspace are converted 'AtBeginDocument' by the next function FBget_glue into tables of three values which are fractions of \fontdimen2, \fontdimen3 and \fontdimen4. If parsing fails due to unexpected syntax, the function returns *nil* instead of a table.

```
function FBget_glue(toks)
  local t = nil
  local f = string.match(toks,
                         "[^%w]hskip%s*([%d%.]*)%s*[^%w]fontdimen 2")
  if f == "" then f = 1 end
  if tonumber(f) then
     t = {tonumber(f), 0, 0}
     f = string.match(toks,    "plus%s*([%d%.]*)%s*[^%w]fontdimen 3")
     if f == "" then f = 1 end
     if tonumber(f) then
        t[2] = tonumber(f)
        f = string.match(toks, "minus%s*([%d%.]*)%s*[^%w]fontdimen 4")
        if f == "" then f = 1 end
        if tonumber(f) then
           t[3] = tonumber(f)
        end
     end
  elseif string.match(toks, "[^%w]F?B?thinspace") then
     t = {0.5, 0, 0}
  elseif string.match(toks, "[^%w]space") then
```

```
    t = {1, 1, 1}
  end
  return t
end
```

Let's initialize the global LuaTeX table FBsp: it holds the characteristics of the glues used in French for high punctuation and quotes and the corresponding no-breaking space characters for option UnicodeNoBreakSpaces.

```
FBsp = {}
FBsp.thin = {}
FBsp.thin.gl = {.5,  0,  0}
FBsp.thin.ch = 0x202F
FBsp.colon = {}
FBsp.colon.gl = {1,  1,  1}
FBsp.colon.ch = 0xA0
FBsp.guill = {}
FBsp.guill.gl = {.8, .3, .8}
FBsp.guill.ch = 0xA0
```

The next function converts the glue table returned by function FBget_glue into sp for the current font; beware of null values for fid, see \nullfont in TikZ, and of special fonts like lcircle1.pfb for which font.getfont(fid) does not return a proper font table, in such cases the function returns nil. \spacekip, when not null, replaces the inter-word space (in raggedright env. f.i.). This is now taken into account.

```
local font_table = {}
local function new_glue_scaled (fid,table)
  if fid > 0 and table[1] then
     local fp = font_table[fid]
     if not fp then
        local ft = font.getfont(fid)
        if ft then
           font_table[fid] = ft.parameters
           fp = font_table[fid]
        end
     end
     local gl = new_node(GLUE,0)
```

\spaceskip is usually 0. In some circumstances (raggedright…) it gets > 0 and then replaces the inter-word space.

```
     if fp then
        local spaceskip
        spaceskip = tex.get(tex.spaceskip)
        local spskip = spaceskip.width
        if spaceskip and spskip and spskip > 0 then
           node.setglue(gl, table[1]*spskip , 0, 0)
```

```
241        else
242           node.setglue(gl, table[1]*fp.space,
243                            table[2]*fp.space_stretch,
244                            table[3]*fp.space_shrink)
245        end
246        return gl
247     else
248        return nil
249     end
250  else
251     return nil
252  end
253 end
```

Let's catch LuaTeX attributes \FB@spacing, \FB@addDPspace and \FB@addGUILspace.

```
254 local FBspacing    = luatexbase.attributes['FB@spacing']
255 local addDPspace   = luatexbase.attributes['FB@addDPspace']
256 local addGUILspace = luatexbase.attributes['FB@addGUILspace']
257 local FBucsNBSP    = luatexbase.attributes['FB@ucsNBSP']
258 local has_attribute = node.has_attribute
```

The following function will be added to kerning callback. It catches all nodes of type GLYPH in the list starting at head and checks the language attributes of the current glyph: nothing is done if the current language is not French and only specific punctuation characters (those for which FB_punct_left or FB_punct_right is true) need a special treatment. In French, local variables are defined to hold the properties of the current glyph (item) and of the previous one (prev) or the next one (next). The FR constant (french) is defined by command \activate@luacode.

```
259 -- Main function (to be added to the kerning callback).
260 local function french_punctuation (head)
```

Restore the built-in kerning for 8-bits fonts.

```
261  node.kerning(head)
262  for item in node.traverse_id(GLYPH, head) do
263    local lang = item.lang
264    local char = item.char
```

Skip glyphs not concerned by French kernings.

```
265    if lang == FR and (FB_punct_left[char] or FB_punct_right[char]) then
266       local fid  = item.font
267       local attr = item.attr
268       local FRspacing = has_attribute(item, FBspacing)
269       FRspacing = FRspacing and FRspacing > 0
270       local FRucsNBSP = has_attribute(item, FBucsNBSP)
271       FRucsNBSP = FRucsNBSP and FRucsNBSP > 0
```

```
      if FRspacing and fid > 0 then
         if FB_punct_left[char] then
            local prev = item.prev
            local prev_id, prev_subtype, prev_char
            if prev then
               prev_id = prev.id
               prev_subtype = prev.subtype
               if prev_id == GLYPH then
                  prev_char = prev.char
               end
            end
```

If the previous node is a glue, check its natural width, only positive glues (actually glues > 1 sp, for tabular 'l' columns) are to be replaced by a non-breaking space.

```
            local is_glue = prev_id == GLUE
            local glue_wd
            if is_glue then
               glue_wd = prev.width
            end
            local realglue = is_glue and glue_wd > 1
```

For characters for which FB_punct_thin or FB_punct_thick is *true*, the amount of spacing to be typeset before them is controlled by commands \FBthinspace and \FBcolonspace respectively. Two options: if a space has been typed in before (turned into *glue* in the node list), we remove the *glue* and add a nobreak penalty and the required *glue*. Otherwise (auto option), the penalty and the required *glue* are inserted if attribute \FB@addDPspace is set, unless any of these four conditions is met: a) node is ':' and the next one is of type GLYPH (avoids spurious spaces in http://mysite, C:\ or 10:35); b) the previous character is part of type FB_punct_null (avoids spurious spaces in strings like (!) or ??); c) a null glue (actually <= 1 sp for tabulars, possibly < 0) preceeds the punctuation character (for tabulars and listings); d) the punctuation character starts a paragraph or an \hbox{}.

When option UnicodeNoBreakSpaces is set to true, a Unicode character U+00A0 or U+202F is inserted instead of penalty and glue.

```
            if FB_punct_thin[char] or FB_punct_thick[char] then
               local SBDP = has_attribute(item, addDPspace)
               local auto = SBDP and SBDP > 0
               if FB_punct_thick[char] and auto then
                  local next = item.next
                  local next_id
                  if next then
                     next_id = next.id
                  end
                  if next_id and
                     (next_id == GLYPH or next_id == DISC) then
```

```
                    auto = false
                end
            end
            if auto then
                if (prev_char and FB_punct_null[prev_char]) or
                    (is_glue and glue_wd <= 1) or
                    (prev_id == HLIST and prev_subtype == 3) or
                    (prev_id == TEMP) then
                    auto = false
                end
            end
            local fbglue
            local t
            if FB_punct_thick[char] then
                  t = FBsp.colon.gl
                  nbspace.char = FBsp.colon.ch
            else
                  t = FBsp.thin.gl
                  nbspace.char = FBsp.thin.ch
            end
            fbglue = new_glue_scaled(fid, t)
```

In case new_glue_scaled fails (returns nil) the node list remains unchanged.

```
            if (realglue or auto) and fbglue then
                if realglue then
                    head = remove_node(head,prev,true)
                end
                if (FRucsNBSP) then
                    nbspace.font = fid
                    nbspace.attr = attr
                    insert_node_before(head,item,copy_node(nbspace))
                else
                    nobreak.attr = attr
                    fbglue.attr  = attr
                    insert_node_before(head,item,copy_node(nobreak))
                    insert_node_before(head,item,copy_node(fbglue))
                end
            end
```

Let's consider '»' and '›' now (the only remaining glyphs of FB_punct_left class): we just have to remove any *glue* possibly preceeding them, then to insert the nobreak penalty and the proper *glue* (controlled by \FBguillspace). If either a) the preceding glyph is member of FB_guil_null, or b) '»'/'›' is the first glyph of an \hbox{} or a paragraph, nothing is done, this is controlled by the addgl flag.

```
        else
            local addgl = (prev_char and
```

```
                                not FB_guil_null[prev_char])
                               or
                               (not prev_char and
                                prev_id ~= TEMP and
                                not (prev_id == HLIST and
                                     prev_subtype == 3)
                               )
```

Correction for tabular ‘c’ (glue 0 plus 1 fil) and ‘l’ (glue 1sp) columns:

```
               if is_glue and glue_wd <= 1 then
                  addgl = false
               end
               local t = FBsp.guill.gl
               nbspace.char = FBsp.guill.ch
               local fbglue = new_glue_scaled(fid, t)
               if addgl and fbglue then
                  if is_glue then
                     head = remove_node(head,prev,true)
                  end
                  if (FRucsNBSP) then
                     nbspace.font = fid
                     nbspace.attr = attr
                     insert_node_before(head,item,copy_node(nbspace))
                  else
                     nobreak.attr = attr
                     fbglue.attr  = attr
                     insert_node_before(head,item,copy_node(nobreak))
                     insert_node_before(head,item,copy_node(fbglue))
                  end
               end
            end
```

Similarly, for ‘«’ or ‘‹’ (unique members of the `FB_punct_right` class): unless either a) the next glyph is member of `FB_guil_null`, or b) ‘«/‹’ is the last glyph of an `\hbox{}` or a paragraph (then the `addgl` flag is false, nothing is done), we remove any *glue* possibly following it and insert first the proper *glue* then a nobreak penalty so that finally the penalty preceeds the *glue*.

```
         elseif FB_punct_right[char] then
            local next = item.next
            local next_id, next_subtype, next_char, nextnext, kern_wd
            if next then
               next_id = next.id
               next_subtype = next.subtype
```

In case of coding «~ or ‹~ remove the penalty and the glue:

```
            if next_id == PENALTY then
               nextnext = next.next
               if nextnext and nextnext.id == GLUE then
                  head = remove_node(head,nextnext,true)
                  head = remove_node(head,next,true)
                  next = item.next
                  if next then
                     next_id = next.id
                     next_subtype = next.subtype
                     if next_id == GLYPH then
                        next_char = next.char
                     end
                  end
               end
            end
```

A kern0 might hide a penalty and/or glue, so look ahead if next is a kern (this occurs with « \texttt{a} » and «~\texttt{a}~»):

```
            if next_id == KERN then
               kern_wd = next.kern
               if kern_wd == 0 then
                  nextnext = next.next
                  if nextnext then
                     next = nextnext
                     next_id = nextnext.id
                     next_subtype = nextnext.subtype
                     if next_id == PENALTY then
                        nextnext = next.next
                        if nextnext and nextnext.id == GLUE then
                           head = remove_node(head,next,true)
                           head = remove_node(head,nextnext,true)
                           next = item.next
                           if next then
                              next_id = next.id
                              next_subtype = next.subtype
                           end
                        end
                     end
                  end
               end
            end
            if next_id == GLYPH then
               next_char = next.char
            end
         end
         local is_glue = next_id == GLUE
```

```
            if is_glue then
               glue_wd = next.width
            end
```

The addgl flag only depends on next_char and is_glue:

```
            local addgl = (next_char and not FB_guil_null[next_char])
                          or (next and not next_char)
```

Correction for tabular ‘c’ columns. For ‘r’ columns, a final ‘«’ character needs to be coded as \mbox{«} for proper spacing (\NoAutoSpacing is another option).

```
            if is_glue and glue_wd == 0 then
               addgl = false
            end
            local fid = item.font
            local t = FBsp.guill.gl
            nbspace.char = FBsp.guill.ch
            local fbglue = new_glue_scaled(fid, t)
            if addgl and fbglue then
               if is_glue then
                  head = remove_node(head,next,true)
               end
               if (FRucsNBSP) then
                  nbspace.font = fid
                  nbspace.attr = attr
                  insert_node_after(head, item, copy_node(nbspace))
               else
                  nobreak.attr = attr
                  fbglue.attr  = attr
                  insert_node_after(head, item, copy_node(fbglue))
                  insert_node_after(head, item, copy_node(nobreak))
               end
            end
         end
      end
   end
end
return head
end
```

This function deals with hyphenation of the euphonic-t in French: strings like “a-t-il”, “dira-t-elle”, “va-t-on”, “semble-t-il”, etc. may be hypenated on the first ‘-’, never on the second one. It increases the hyphen penalty to 10000 on the second ‘-’.

```
local FB_t =
 {[0x74]             = true,
  [0x54]             = true}
local function euphonic_t (head)
```

```
  for item in node.traverse_id(DISC, head) do
    if item.subtype == 2 then
       local next = item.next
       local lang
       local nnext
       if next and next.id == GLYPH and FB_t[next.char] then
          lang = next.lang
          nnext = next.next
       end
       if lang == FR and nnext and
          nnext.id == DISC and nnext.subtype == 2 then
          nnext.penalty = 10000
       end
    end
  end
  return head
end
return french_punctuation, euphonic_t
</lua>
```

As a language tag is part of glyph nodes in LuaTeX, no more switching has to be done in \extrasfrench.

The next definition will be used to activate Lua punctuation: it loads frenchb.lua and adds function french_punctuation to the kerning callback; "adding" anything actually disables the built-in kerning for Type1 fonts (which is now added to french_punctuation).

```
<*french>
\def\activate@luacode{%
    \directlua{%
      FR = \the\l@french ;
      local path = kpse.find_file("frenchb.lua", "lua")
      if path then
         local f1, f2 = dofile(path)
         luatexbase.add_to_callback("kerning",
                    f1, "frenchb.french_punctuation")
         luatexbase.add_to_callback("pre_linebreak_filter",
                    f2, "frenchb.euphonic_t")
      else
         texio.write_nl('')
         texio.write_nl('*****************************')
         texio.write_nl('Error: frenchb.lua not found.')
         texio.write_nl('*****************************')
         texio.write_nl('')
      end
     }%
}
```

A new 'if' \ifFBAutoSpacePunctuation needs to be defined now to control the two possible ways of dealing with 'high punctuation'. it's default value is true, but it can be set to false by \frenchsetup{AutoSpacePunctuation=false} for finer control.

```
\newif\ifFBAutoSpacePunctuation  \FBAutoSpacePunctuationtrue
```

\AutoSpaceBeforeFDP
\NoAutoSpaceBeforeFDP

\autospace@beforeFDP and \noautospace@beforeFDP are internal commands. \autospace@beforeFDP sets LuaTeX attribute \FB@addDPspace to 1 (true), while \noautospace@beforeFDP sets flag \FB@addDPspace to 0 (false). User commands \AutoSpaceBeforeFDP and \NoAutoSpaceBeforeFDP do the same and take care of the flag \ifFBAutoSpacePunctuation in LaTeX.

Set the default now for Plain (done later for LaTeX).

```
\def\autospace@beforeFDP{\FB@addDPspace=\@ne \relax}
\def\noautospace@beforeFDP{\FB@addDPspace=\z@ \relax}
\ifLaTeXe
  \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
                          \FBAutoSpacePunctuationtrue}
  \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
                            \FBAutoSpacePunctuationfalse}
  \AtEndOfPackage{\AutoSpaceBeforeFDP}
\else
  \let\AutoSpaceBeforeFDP\autospace@beforeFDP
  \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
  \AutoSpaceBeforeFDP
\fi
```

\NoAutoSpacing

The following command disables automatic spacing for high punctuation and French quote characters. It is meant to be used inside a group. \NoAutoSpacing must be inhibited in bookmarks. The faked definition of \texorpdfstring will be overwritten by hyperref.sty.

```
\providecommand\texorpdfstring[2]{#1}
\DeclareRobustCommand{\NoAutoSpacing}{%
              \texorpdfstring{\FB@spacing=\z@}{}%
}
```

2.3 Commands for French quotation marks

\guillemotleft
\guillemotright
\guilsinglleft
\guilsinglright
\textquoteddblleft
\textquoteddblright

We provide the following definitions for non-LaTeX users only as fall-back, they are welcome to change them for anything better.

```
\ifLaTeXe
\else
  \def\guillemotleft{{\char"00AB}}
  \def\guillemotright{{\char"00BB}}
  \def\textquotedblleft{{\char"201C}}
  \def\textquotedblright{{\char"201D}}
```

```
  \def\guilsinglleft{{\char"2029}}
  \def\guilsinglright{{\char"203A}}
  \let\xspace\relax
\fi
```

\og \fg The user level macros for quotation marks are named \og (“ouvrez guillemets’’) and \fg (“fermez guillemets’’). They are kept for backward compatibility only, as typing in « and » is much easier. Another option for typesetting quotes in French is to use the command \frquote (see below). If the current language is not French, \og and \fg provide default (English) quotes.

```
\newcommand*{\og}{\textquotedblleft}
\newcommand*{\fg}{%
  \texorpdfstring{\ifdim\lastskip>\z@\unskip\fi\textquotedblright\xspace}%
                 {\textquotedblright\xspace}%
  }
```

The definitions of \og and \fg for quotation marks are switched on and off through the \extrasfrench \noextrasfrench mechanism. Outside French, \og and \fg will typeset standard English opening and closing double quotes. We’ll try to be smart to users of David Carlisle’s xspace package: if this package is loaded there will be no need for {} or \ to get a space after \fg, otherwise \xspace will be defined as \relax (done at the end of this file).

```
\newcommand*{\FB@og}{\texorpdfstring{\guillemotleft}%
                                    {\guillemotleft\space}}
\newcommand*{\FB@fg}{\texorpdfstring{\ifdim\lastskip>\z@\unskip\fi
                             \guillemotright}{\space\guillemotright}}
\def\bbl@frenchguillemets{\def\og{\FB@og}%
                          \def\fg{\FB@fg\xspace}}
\addto\extrasfrench{\babel@save\og \babel@save\fg
                    \bbl@frenchguillemets}
```

\frquote Another way of entering French quotes relies on \frquote{} with supports up to two levels of quotes. Let’s define the default quote characters to be used for level one or two of quotes…

```
\newcommand*{\@ogi}{\ifmmode\hbox{\guillemotleft}\else\guillemotleft\fi}
\newcommand*{\@fgi}{\ifmmode\hbox{\guillemotright}\else\guillemotright\fi}
\newcommand*{\ogii}{\ifFBInnerGuillSingle \guilsinglleft
                    \else                 \textquotedblleft
                    \fi}
\newcommand*{\fgii}{\ifFBInnerGuillSingle \guilsinglright
                    \else                 \textquotedblright
                    \fi}
\newcommand*{\@ogii}{\ifmmode\hbox{\ogii}\else\ogii\fi}
\newcommand*{\@fgii}{\ifmmode\hbox{\fgii}\else\fgii\fi}
```

and the needed technical stuff to handle options:

```
\newcount\FBguill@level
\newtoks\FBold@everypar
```

For PlainTeX, the definition of `\FB@addquote@everypar` is borrowed from `csquotes.sty`.

```
\ifLaTeXe
  \def\FB@addquote@everypar{%
    \ifx\FBeveryparguill\FBguillnone
    \else
      \AddToHook{para/begin}[.]%
          {\box\IndentBox\FBeverypar@quote\OmitIndent}%
    \fi}
  \def\FB@removequote@everypar{%
    \ifx\FBeveryparguill\FBguillnone
    \else \RemoveFromHook{para/begin}[.]%
    \fi}
\else
  \def\FB@addquote@everypar{%
    \let\FBnew@everypar\everypar
    \FBold@everypar=\expandafter{\the\everypar}%
    \FBnew@everypar={\the\FBold@everypar\FBeverypar@quote}%
    \let\everypar\FBold@everypar
    \let\FB@addquote@everypar\relax
  }
  \let\FB@removequote@everypar\relax
\fi
\newif\ifFBcloseguill  \FBcloseguilltrue
\newif\ifFBInnerGuillSingle
\def\FBguillopen{\guillemotleft}
\def\FBguillclose{\guillemotright}
\let\FBguillnone\empty
\let\FBeveryparguill\FBguillopen
\let\FBeverylineguill\FBguillnone
\let\FBeverypar@quote\relax
\let\FBeveryline@quote\empty
```

The main command `\frquote` accepts (in LaTeX2e only) a starred version which suppresses the closing quote; it is meant to be used for inner quotations which end together with the outer one, then only one closing guillemet (the outer one) should be printed. `\frquote` (without star) is now designed to work in bookmarks too.

```
\ifLaTeXe
  \DeclareRobustCommand\frquote{%
    \texorpdfstring{\@ifstar{\FBcloseguillfalse\fr@quote}%
                            {\FBcloseguilltrue \fr@quote}}%
                   {\bm@fr@quote}%
  }
  \newcommand{\bm@fr@quote}[1]{« #1 »}
```

```
\else
  \newcommand\frquote[1]{\fr@quote{#1}}
\fi
```

The internal command \fr@quote takes one (long) argument: the quotation text.

```
\newcommand{\fr@quote}[1]{%
  \leavevmode
  \advance\FBguill@level by \@ne
  \ifcase\FBguill@level
  \or
```

This for level 1 (outer) quotations: set \FBeverypar@quote for level 1 quotations and add it at each paragraph start using \FB@addquote@everypar, then print the quotation:

```
    \ifx\FBeveryparguill\FBguillnone
      \let\FBeverypar@quote\relax
    \else
      \def\FBeverypar@quote{\FBeveryparguill}%
      \FB@addquote@everypar
    \fi
    \@ogi #1\@fgi
  \or
```

This for level 2 (inner) quotations: Omega's command \localleftbox included in LuaTeX, is convenient for repeating guillemets at the beginning of every line.

```
    \ifx\FBeverylineguill\FBguillopen
      \def\FBeveryline@quote{\guillemotleft\FBguillspace}%
      \localleftbox{\FBeveryline@quote}%
      \let\FBeverypar@quote\relax
      \@ogi #1\ifFBcloseguill\@fgi\fi
    \else
      \ifx\FBeverylineguill\FBguillclose
        \def\FBeveryline@quote{\guillemotright\FBguillspace}%
        \localleftbox{\FBeveryline@quote}%
        \let\FBeverypar@quote\relax
        \@ogi #1\ifFBcloseguill\@fgi\fi
      \else
```

otherwise we eventually need to redefine \FBeverypar@quote for level 2 quotations:

```
        \let\FBeverypar@quote\relax
        \ifFBInnerGuillSingle
          \ifx\FBeveryparguill\FBguillopen
            \def\FBeverypar@quote{\guilsinglleft\FBguillspace}%
          \fi
          \ifx\FBeveryparguill\FBguillclose
            \def\FBeverypar@quote{\guilsinglright\FBguillspace}%
```

```
          \fi
        \fi
        \@ogii #1\@fgii
      \fi
    \fi
  \else
```

Warn if \FBguill@level > 2:

```
    \ifx\PackageWarning\@undefined
      \fb@warning{\noexpand\frquote\space handles up to
                  two levels.\\ Quotation not printed.}%
    \else
      \PackageWarning{french.ldf}{%
         \protect\frquote\space handles up to two levels.
         \MessageBreak Quotation not printed.  Reported}
    \fi
  \fi
```

Closing: step down \FBguill@level and clean on exit. Changes made global in case \frquote{} ends inside an environment.

```
  \global\advance\FBguill@level by \m@ne
  \ifcase\FBguill@level \global\let\FBeverypar@quote\relax
      \FB@removequote@everypar
  \or \gdef\FBeverypar@quote{\FBeveryparguill}%
      \global\let\FBeveryline@quote\empty
      \ifx\FBeverylineguill\FBguillnone\else\localleftbox{}\fi
  \fi
}
```

The next command is intended to be used in list environments to suppress quotes which might be added by \FBeverypar@quote after items for instance.

```
\newcommand*{\NoEveryParQuote}{%
  \let\FBeveryparguill\FBguillnone
  \let\FBeverypar@quote\relax
}
```

2.4 Date in French

\frenchtoday
\frenchdate
\datefrench

The following code creates a macro \datefrench which in turn defines command \frenchtoday (\today is defined as \frenchtoday in French). This new implementation relies on commands \SetString and \SetStringLoop, therefore requires Babel 3.10 or newer.

```
\StartBabelCommands*{french}{date}
    [unicode, fontenc=TU EU1 EU2, charset=utf8]
    \SetStringLoop{month#1name}{%
```

```
      janvier,février,mars,avril,mai,juin,juillet,%
      août,septembre,octobre,novembre,décembre}
    \SetString\today{\FB@date{\year}{\month}{\day}}
\EndBabelCommands
```

\frenchdate (which produces an unbreakable string) and \frenchtoday (breakable) both rely on \FB@date, the inner group is needed for \hbox.

```
\newcommand*{\FB@date}[3]{%
  {{\number#3}\ifnum1=#3{\ier}\fi\FBdatespace
  \csname month\romannumeral#2name\endcsname
  \ifx#1\@empty\else\FBdatespace\number#1\fi}}
\newcommand*{\FBdatebox}{\hbox}
\newcommand*{\FBdatespace}{\space}
\newcommand*{\frenchdate}{\FBdatebox\FB@date}
```

2.5 Extra utilities

Let's provide the French user with some extra utilities.

\up \up eases the typesetting of superscripts like '1er'.

\fup When a font has built-in superscripts, the best thing to do is to just use them, otherwise \fup provides an alternative which typsesets superscripts slightly smaller and higher. Scaling is done using package scalefnt which will be loaded at the end of Babel's loading (babel-french being an option of Babel, it cannot load a package while being read).

Options FrenchSuperscripts and LowercaseSuperscripts will be processed in \FBprocess@options to choose which version of \up{} will be used in the document.

```
\newif\ifFBFrenchSuperscripts       \FBFrenchSuperscriptstrue
\newif\ifFBLowercaseSuperscripts    \FBLowercaseSuperscriptstrue
\newdimen\FB@Mht
\ifLaTeXe
  \AtEndOfPackage{\RequirePackage{scalefnt}}
```

\fup holds the definition of fake superscripts. The scaling ratio is 0.65, raising is computed to put the top of lower case letters (like 'm') just under the top of upper case letters (like 'M'), precisely 12% down. These settings look correct for most fonts, but can be tuned by the end-user if necessary by changing \FBsupR and \FBsupS commands.

\FB@lc is defines as \MakeLowercase to inhibit the uppercasing of superscripts (this may happen in page headers with the standard classes but is wrong); \FB@lc can be redefined to do nothing using option LowercaseSuperscripts=false of \frenchsetup{}.

```
  \newcommand*{\FBsupR}{-0.12}
  \newcommand*{\FBsupS}{0.65}
  \newcommand*{\FB@lc}[1]{\MakeLowercase{#1}}
```

```
\NewDocumentCommand\fup{ m }{%
  \settoheight{\FB@Mht}{M}%
  \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
  \addtolength{\FB@Mht}{-\FBsupS ex}%
  \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
  }
```

Poor man's definition of \up for Plain.

```
\else
  \providecommand*{\up}[1]{\leavevmode\raise1ex\hbox{\sevenrm #1}}
\fi
```

\ieme \ier \iere \iemes \iers \ieres

Some handy macros for those who don't know how to abbreviate ordinals:

```
\def\ieme{\up{e}\xspace}
\def\iemes{\up{es}\xspace}
\def\ier{\up{er}\xspace}
\def\iers{\up{ers}\xspace}
\def\iere{\up{re}\xspace}
\def\ieres{\up{res}\xspace}
```

\FBmedkern \FBthickkern

Configurable kerns \FBmedkern, and \FBthickkern suitable for HTML translation.

```
\newcommand*{\FBmedkern}{\kern+.2em}
\newcommand*{\FBthickkern}{\kern+.3em}
```

\primo \fprimo) \nos \Nos \No \no

Some support macros relying on \up for numbering, safe in bookmarks:

```
\newcommand*{\FrenchEnumerate}[1]{%
    #1\texorpdfstring{\up{o}\FBthickkern}{\textdegree\space}}
\newcommand*{\FrenchPopularEnumerate}[1]{%
    #1\texorpdfstring{\up{o})\FBthickkern}{\textdegree\space}}
```

Typing \primo should result in '^{o} ' (except in bookmarks where \textdegree is used instead of o-superior),

```
\def\primo{\FrenchEnumerate1}
\def\secundo{\FrenchEnumerate2}
\def\tertio{\FrenchEnumerate3}
\def\quarto{\FrenchEnumerate4}
```

while typing \fprimo) gives '^{o}) (except in bookmarks where \textdegree is used instead),.

```
\def\fprimo){\FrenchPopularEnumerate1}
\def\fsecundo){\FrenchPopularEnumerate2}
\def\ftertio){\FrenchPopularEnumerate3}
\def\fquarto){\FrenchPopularEnumerate4}
```

Let's provide four macros for the common abbreviations of "Numéro". In bookmarks ° is used instead of o-superior.

```
\DeclareRobustCommand*{\No}{%
    \texorpdfstring{N\up{o}\FBmedkern}{N\textdegree\space}}
```

```
\DeclareRobustCommand*{\no}{%
    \texorpdfstring{n\up{o}\FBmedkern}{n\textdegree\space}}
\DeclareRobustCommand*{\Nos}{%
    \texorpdfstring{N\up{os}\FBmedkern}{N\textdegree\space}}
\DeclareRobustCommand*{\nos}{%
    \texorpdfstring{n\up{os}\FBmedkern}{n\textdegree\space}}
```

\bname
\bsc

These commands are meant to easily enter family names (in small capitals for the latter) while avoidind hyphenation. A \kern0pt is used instead of \mbox because \mbox would break microtype's font expansion; as a positive side effect, composed names (such as Dupont-Durand) can now be hyphenated on explicit hyphens.

```
\ifLaTeXe
  \DeclareRobustCommand*{\bname}[1]{%
    \texorpdfstring{\leavevmode\begingroup\kern0pt #1\endgroup}{#1}%
  }
  \DeclareRobustCommand*{\bsc}[1]{%
    \texorpdfstring{\leavevmode\begingroup\kern0pt \scshape #1\endgroup}%
                   {\textsc{#1}}%
  }
\else
  \newcommand*{\bname}[1]{\leavevmode\begingroup\kern0pt #1\endgroup}
  \let\bsc\bname
\fi
```

Some definitions for special characters. We won't define \tilde as a Text Symbol not to conflict with the macro \tilde for math mode and use the name \tild instead. Note that \boi may *not* be used in math mode, its name in math mode is \backslash.

```
\providecommand*{\textbackslash}{{\char"005C}}
\providecommand*{\textasciicircum}{{\char"005E}}
\providecommand*{\textasciitilde}{{\char"007E}}
\providecommand*{\degre}{°}
\providecommand*{\degres}{°}
\providecommand*{\boi}{\textbackslash}
\providecommand*{\circonflexe}{\textasciicircum}
\providecommand*{\tild}{\textasciitilde}
\providecommand*{\at}{@}
```

2.6 Formatting numbers

\StandardMathComma
\DecimalMathComma

As mentioned in the TEXbook p. 134, the comma is of type \mathpunct in math mode: it is automatically followed by a thin space. This is convenient in lists and intervals but unpleasant when the comma is used as a decimal separator in French: it has to be entered as {,}. \DecimalMathComma makes the comma be an ordinary character (of type \mathord) in French *only* (no space added); \StandardMathComma switches back to the standard behaviour of the comma.

Unfortunately, \newcount inside \if breaks Plain formats.

```
\newif\ifFB@icomma
\newcount\mc@charclass
\newcount\mc@charfam
\newcount\mc@charslot
\newcount\std@mcc
\newcount\dec@mcc
  \mc@charclass=\Umathcharclass`\,
  \newcommand*{\dec@math@comma}{%
    \mc@charfam=\Umathcharfam`\,
    \mc@charslot=\Umathcharslot`\,
    \Umathcode`\,= 0 \mc@charfam \mc@charslot
  }
  \newcommand*{\std@math@comma}{%
    \mc@charfam=\Umathcharfam`\,
    \mc@charslot=\Umathcharslot`\,
    \Umathcode`\,= \mc@charclass \mc@charfam \mc@charslot
  }
\let\dec@m@c\relax
```

If \DecimalMathComma is issued in the document body (when the current language is French) its effect will survive to a language switch, unless issued inside a group (see \dec@m@c's expansion). The icomma inhibits \DecimalMathComma.

```
\newif\if@FBpreamble
\ifLaTeXe \@FBpreambletrue \fi
\newif\if@preamble@DecimalMathComma
\newcommand*{\DecimalMathComma}{%
  \if@FBpreamble \@preamble@DecimalMathCommatrue
  \else
    \ifFB@icomma
      \PackageWarning{french.ldf}{%
        icomma package loaded, \protect\DecimalMathComma\MessageBreak
        does nothing.  Reported}%
    \else
      \ifFBfrench
        \dec@math@comma
        \let\dec@m@c\dec@math@comma
        \expandafter\addto\csname extras\languagename\endcsname
          {\dec@m@c}%
      \fi
    \fi
  \fi
}
\newcommand*{\StandardMathComma}{%
  \ifFB@icomma
    \PackageWarning{french.ldf}{%
      icomma package loaded, \protect\StandardMathComma\MessageBreak
```

```
      does nothing.  Reported}%
  \else
    \ifFBfrench
      \std@math@comma
      \let\dec@m@c\relax
    \fi
  \fi
}
```

This is for Plain formats *only* (see below).

```
\ifLaTeXe\else
  \addto\noextrasfrench{\std@math@comma}
\fi
```

Fake command `\nombre` for Plain based formats, warning users of `babel-french` v. 1.x. about the change:

```
\newcommand*{\nombre}[1]{{#1}\fb@warning{*** \noexpand\nombre
                                  no longer formats numbers\string! ***}}
```

Let's activate LuaTeX punctuation if necessary (LaTeX or Plain) so that `\FBsetspaces` commands can be used in the preamble, then cleanup and exit without loading any `.cfg` file in case of Plain formats.

```
\activate@luacode
\let\FBstop@here\relax
\def\FBclean@on@exit{%
  \let\ifLaTeXe\iffalse
  \let\LaTeXetrue\undefined
  \let\LaTeXefalse\undefined
  \let\FB@llc\loadlocalcfg
  \let\loadlocalcfg\@gobble}
\ifx\magnification\@undefined
\else
  \def\FBstop@here{%
    \FBclean@on@exit
    \ldf@finish\CurrentOption
    \let\loadlocalcfg\FB@llc
    \endinput}
\fi
\FBstop@here
```

What follows is for LaTeX2e *only*: the next piece of code would break Plain formats. If issued in the preamble, `\DecimalMathComma` works globally on all parts of the document that are typeset in French. Can be canceled anytime by `\StandardMathComma`.

```
\AddToHookNext{env/document/before}{%
  \@FBpreamblefalse
  \IfPackageLoadedTF{icomma}%
```

```
    {\FB@icommatrue
     \if@preamble@DecimalMathComma
       \FBWarning{icomma package loaded, \protect\DecimalMathComma%
                  \MessageBreak does nothing.  Reported}%
     \fi
    }%
    {\if@preamble@DecimalMathComma
       \ifFB@mainlanguage@FR \dec@math@comma \fi
       \let\dec@m@c\dec@math@comma
       \addto\extrasfrench{\dec@m@c}%
     \fi
```

The comma is reset to type \mathpunct when leaving French (only if the icomma package is not loaded).

```
     \addto\noextrasfrench{\std@math@comma}%
    }%
}
```

nombre We redefine \nombre for LaTeX2e. The command \nombre is now borrowed from numprint.sty for LaTeX2e. There is no point to maintain the former tricky code when a package is dedicated to do the same job and more. A warning is issued at the first call of \nombre if \numprint is not defined, suggesting what to do. The package numprint is *not* loaded automatically by babel-french because of possible options conflict.

```
\renewcommand*{\nombre}[1]{\Warning@nombre{#1}}
\newcommand*{\Warning@nombre}[1]{%
  \ifdefined\numprint
    \numprint{#1}%
  \else
    \FBWarning{%
       \protect\nombre\space now relies on package numprint.sty,%
       \MessageBreak add \protect
       \usepackage[autolanguage]{numprint},\MessageBreak
       see file numprint.pdf for more options.\MessageBreak
       \protect\nombre\space called}%
    \global\let\Warning@nombre\relax
    {#1}%
  \fi
}

\newcommand*{\FBthousandsep}{\kern \fontdimen2\font \relax}
```

2.7 Caption names

The next step consists in defining the French equivalents for the LaTeX caption names.

New implementation for caption names (requires Babel's 3.10 or newer).

```
\StartBabelCommands*{french}{captions}
      [unicode, fontenc=TU EU1 EU2, charset=utf8]
  \SetString{\refname}{Références}
  \SetString{\abstractname}{Résumé}
  \SetString{\bibname}{Bibliographie}
  \SetString{\chaptername}{Chapitre}
  \SetString{\prefacename}{Préface}
  \SetString{\appendixname}{Annexe}
  \SetString{\contentsname}{Table des matières}
  \SetString{\listfigurename}{Table des figures}
  \SetString{\listtablename}{Liste des tableaux}
  \SetString{\indexname}{Index}
  \SetString{\glossaryname}{Glossaire}
  \SetString{\figurename}{Figure}
  \SetString{\tablename}{Table}
  \SetString{\pagename}{page}
  \SetString{\seename}{voir}
  \SetString{\alsoname}{voir aussi}
  \SetString{\enclname}{P.~J. }
  \SetString{\ccname}{Copie à }
  \SetString{\headtoname}{}
  \SetString{\proofname}{Démonstration}
  \SetString{\partnameord}{partie}
  \SetString{\partfirst}{Première}
  \SetString{\partsecond}{Deuxième}
```

When `PartNameFull=true` (default), `\part{}` is printed in French as "Première partie'' instead of "Partie I''. As logic is prohibited inside `\SetString`, let's hide the test about `PartNameFull` in `\FB@partname`.

```
 \SetStringLoop{ordinal#1}{%
    \frenchpartfirst,\frenchpartsecond,Troisième,Quatrième,%
    Cinquième,Sixième,Septième,Huitième,Neuvième,Dixième,Onzième,%
    Douzième,Treizième,Quatorzième,Quinzième,Seizième,%
    Dix-septième,Dix-huitième,Dix-neuvième,Vingtième}
  \AfterBabelCommands{%
    \NewDocumentCommand\FB@emptypart{}{\def\thepart{\unskip}}%
    \NewDocumentCommand\FB@partname{}{%
       \ifFBPartNameFull
         \csname ordinal\romannumeral\value{part}\endcsname\space
         \partnameord\FB@emptypart
       \else
         Partie%
       \fi}%
   }
  \SetString{\partname}{\FB@partname}
\EndBabelCommands
```

\figurename and \tablename are printed in small caps in French, unless either SmallCapsFigTabCaptions is set to false or a class or package loaded before babel-french defines \FBfigtabshape as \relax.

```
\providecommand*{\FBfigtabshape}{\scshape}
```

\FBtocpartsep New command \FBtocpartname to help printing “Première partie” instead of “Partie I” in the Table of Contents. It takes a Roman numeral as argument (the part number), and returns a formatted string (“Première partie” if the argument is “I”), unless option TocPartNameFull is set to false. It is used currently used only with the memoir and koma-script classes.

```
\ExplSyntaxOn
\NewExpandableDocumentCommand\FBlower{m}
 {
  \str_lowercase:n {#1}
 }
\ExplSyntaxOff
\newcommand*\FBtocpartsep{\protect\space}
\NewDocumentCommand\FBtocpartname{m}{%
    \ifFBTocPartNameFull
      \csname ordinal\FBlower{#1}\endcsname\space
           \partnameord \FBtocpartsep
    \else
      Partie%
    \fi}%
```

2.8 Checks about packages’ loading order

\FBWarning \FBWarning is an alias of \PackageWarning{french.ldf} which can be made silent by option SuppressWarning.

```
\newcommand{\FBWarning}[1]{\PackageWarning{french.ldf}{#1}}
```

Package beamerarticle should be loaded before babel-french to avoid list’s conflicts, see p. 43.

```
\newif\if@FBwarning@beamerarticle
\IfPackageLoadedTF{beamerarticle}{}{\@FBwarning@beamerarticletrue}
\AddToHookNext{env/document/before}{%
   \if@FBwarning@beamerarticle
     \IfPackageLoadedTF{beamerarticle}{}%
                                      {\@FBwarning@beamerarticlefalse}%
   \fi
   \if@FBwarning@beamerarticle
     \FBWarning{Please load the "beamerarticle" package\MessageBreak
                BEFORE babel/french; reported}%
   \fi
}
```

2.9 Setup options: key/value stuff (l3keys)

Check LaTeX2e version (support for l3keys required).

```
\NeedsTeXFormat{LaTeX2e}[2022-06-01]
```

All setup options are handled by command \frenchsetup{} based on the l3keys' \SetKeys{} command. A list of flags is defined beforehand and set to default values which will possibly be changed 'AtEndOfPackage' in case French is the main language. After this, \frenchsetup{} eventually modifies the preset values of these flags.
Some options processing occurs in \frenchsetup{}, *only for options explicitly set* by \frenchsetup{}, the rest is done just before \begin{document}.
We first define a collection of conditionals for global layout. Their defaults values are chosen so that babel-french does not change anything regarding the global layout. Some of them will toggled 'AtEndofPackage' according to the main language, then they will all be checked again just before \begin{document} in \FBprocess@options to fit \frenchsetup{} specifications and changes required by packages loaded after Babel.

```
\newif\ifFBShowOptions
\newif\ifFBStandardLayout              \FBStandardLayouttrue
\newif\ifFBStandardListSpacing         \FBStandardListSpacingtrue
\newif\ifFBListItemsAsPar
\newif\ifFBNosepItemize                \FBNosepItemizetrue
\newif\ifFBNosepEnumerate
\newif\ifFBStandardItemizeEnv          \FBStandardItemizeEnvtrue
\newif\ifFBStandardEnumerateEnv        \FBStandardEnumerateEnvtrue
\newif\ifFBStandardItemLabels          \FBStandardItemLabelstrue
\newif\ifFBStandardLists               \FBStandardListstrue
\newif\ifFBIndentFirst
\newif\ifFBFrenchFootnotes
\newif\ifFBAutoSpaceFootnotes
\newif\ifFBOriginalTypewriter
\newif\ifFBThinColonSpace
\newif\ifFBThinSpaceInFrenchNumbers
\newif\ifFBUnicodeNoBreakSpaces
\newif\ifFBINGuillSpace
\newif\ifFBPartNameFull
\newif\ifFBTocPartNameFull
\newif\ifFBSmallCapsFigTabCaptions
\newif\ifFBCustomiseFigTabCaptions
\newif\ifFBSuppressWarning
```

If the new templates for lists and footnotes are available, babel-french will use them as far as possible.

```
\newif\ifFBnewlists
\newif\ifFBnewfootnotes
```

```
\IfDocumentMetadataTF
  {\IfFormatAtLeastTF{2026-06-01}%
     {\FBnewliststrue}%
     {\def\warning@FBLaTeXFormat{%
        \FBWarning{%
         LaTeX Format 2026-06-01 is required for lists.\MessageBreak
         Please update your LaTeX distribution\MessageBreak
         or remove the \protect\DocumentMetadata{} command.\MessageBreak
         Otherwise babel-french will NOT customise lists.\MessageBreak
         Reported}%
        }%
     }%
   \FBnewfootnotestrue
  }{}
```

Some specific code for the koma-script classes.

```
\newif\ifFB@koma
\ifLaTeXe
  \IfClassLoadedTF{scrartcl}{\FB@komatrue}{}
  \IfClassLoadedTF{scrbook}{\FB@komatrue}{}
  \IfClassLoadedTF{scrreprt}{\FB@komatrue}{}
\fi
\ifFB@koma
  \ifdefined\partformat
    \def\FB@partformat@fix{%
           \ifFBPartNameFull
             \babel@save\partformat
             \renewcommand*{\partformat}{\partname}%
           \fi}
    \addto\extrasfrench{\FB@partformat@fix}%
  \fi
\fi
```

Some of the flags must be toogled when French is the main language. The latter (last option of Babel, stored in \bbl@main@language) will be known 'AtEndOfPackage'. So we postpone the \bbl@main@language check until then.
Our list customisation conflicts with the beamer class and with the beamerarticle package. The patch provided in beamerbasecompatibility solves the conflict except in case of language changes, so we provide our own patch. When the beamer is loaded, lists are not customised at all to ensure compatibility. The beamerarticle package needs to be loaded *before* Babel, a warning is issued otherwise, see section 2.8; a light customisation is compatible with the beamerarticle package.

```
\def\FB@french{french}
\newif\ifFB@mainlanguage@FR
\AtEndOfPackage{%
  \ifx\bbl@main@language\FB@french \FB@mainlanguage@FRtrue \fi
```

```
  \ifFB@mainlanguage@FR
```

beamer requires special tuning for legacy lists:

```
    \ifFBnewlists
      \FBStandardListSpacingfalse
      \FBStandardItemizeEnvfalse
      \FBStandardEnumerateEnvfalse
      \FBStandardItemLabelsfalse
    \else
      \IfClassLoadedTF{beamer}%
        {\PackageInfo{french.ldf}{%
         No list customisation for the beamer class,%
         \MessageBreak reported}%
        }%
        {\IfPackageLoadedTF{beamerarticle}%
           {\FBStandardItemLabelsfalse
            \FBStandardListSpacingfalse
            \PackageInfo{french.ldf}{%
              Minimal list customisation for the beamerarticle%
              \MessageBreak package; reported}%
           }%
```

Otherwise customise lists “à la française”:

```
           {\FBStandardListSpacingfalse
            \FBStandardItemizeEnvfalse
            \FBStandardEnumerateEnvfalse
            \FBStandardItemLabelsfalse
           }%
        }
    \fi
    \FBIndentFirsttrue
    \FBFrenchFootnotestrue
    \FBAutoSpaceFootnotestrue
    \FBPartNameFulltrue
    \FBTocPartNameFulltrue
    \FBStandardLayouttrue
    \FBSmallCapsFigTabCaptionstrue
  \fi
}
```

\frenchsetup Let’s define the keys to be used in \frenchsetup{}.

```
\DeclareKeys[FBsetup]
{
  ShowOptions.if               = FBShowOptions                        ,
  StandardLayout.default:n     = {true}                               ,
  StandardLayout.code          = \FBStandardLayout@setup{#1}          ,
```

```
    StandardListSpacing.if       = FBStandardListSpacing                ,
    ReduceListSpacing.ifnot      = FBStandardListSpacing                ,
    NosepItemize.if              = FBNosepItemize                       ,
    NosepItemize.default:n       = {true}                               ,
    NosepItemize.code            = \FBNosepItemize@setup{#1}            ,
    NosepEnumerate.if            = FBNosepEnumerate                     ,
    NosepEnumerate.code          = \FBNosepEnumerate@setup{#1}          ,
    StandardItemizeEnv.if        = FBStandardItemizeEnv                 ,
    StandardEnumerateEnv.if      = FBStandardEnumerateEnv               ,
    StandardItemLabels.if        = FBStandardItemLabels                 ,
    ItemLabels.store             = \FrenchLabelItem                     ,
    ItemLabeli.store             = \Frlabelitemi                        ,
    ItemLabelii.store            = \Frlabelitemii                       ,
    ItemLabeliii.store           = \Frlabelitemiii                      ,
    ItemLabeliv.store            = \Frlabelitemiv                       ,
    StandardLists.default:n      = {true}                               ,
    StandardLists.code           = \FBStandardLists@setup{#1}           ,
    ListItemsAsPar.if            = FBListItemsAsPar                     ,
    IndentFirst.if               = FBIndentFirst                        ,
    FrenchFootnotes.if           = FBFrenchFootnotes                    ,
    AutoSpaceFootnotes.if        = FBAutoSpaceFootnotes                 ,
    AutoSpacePunctuation.if      = FBAutoSpacePunctuation               ,
    OriginalTypewriter.if        = FBOriginalTypewriter                 ,
    ThinColonSpace.default:n     = {true}                               ,
    ThinColonSpace.code          = \FBThinColonSpace@setup{#1}          ,
    ThinSpaceInFrenchNumbers.if  = FBThinSpaceInFrenchNumbers           ,
    UnicodeNoBreakSpaces.if      = FBUnicodeNoBreakSpaces               ,
    FrenchSuperscripts.if        = FBFrenchSuperscripts                 ,
    LowercaseSuperscripts.if     = FBLowercaseSuperscripts              ,
    PartNameFull.if              = FBPartNameFull                       ,
    TocPartNameFull.if           = FBTocPartNameFull                    ,
    CustomiseFigTabCaptions.default:n = {true}                          ,
    CustomiseFigTabCaptions.code = \FBCustomiseFigTabCaptions@setup{#1} ,
    SmallCapsFigTabCaptions.default:n = {true}                          ,
    SmallCapsFigTabCaptions.code = \FBSmallCapsFigTabCaptions@setup{#1} ,
    SuppressWarning.default:n    = {true}                               ,
    SuppressWarning.code         = \FBSuppressWarning@setup{#1}         ,
    INGuillSpace.default:n       = {true}                               ,
    INGuillSpace.code            = \FBINGuillSpace@setup{#1}            ,
    InnerGuillSingle.if          = FBInnerGuillSingle                   ,
    EveryParGuill.default:n      = {open}                               ,
    EveryParGuill.code           = \FBEveryParGuill@setup{#1}           ,
    EveryLineGuill.default:n     = {open}                               ,
    EveryLineGuill.code          = \FBEveryLineGuill@setup{#1}          ,
    og.code                      = \FBog@setup{#1}                      ,
    fg.code                      = \FBfg@setup{#1}                      ,
```

```
}
```

Let's now define this command which reads and sets the options to be processed either immediately (i.e. just after setting the key) or just before \begin{document})by \FBprocess@options. \frenchsetup{} can only be called in the preamble.

```
\newcommand*{\frenchsetup}[1]{%
  \SetKeys[FBsetup]{#1}%
}%
\@onlypreamble\frenchsetup
```

Keep the former name \frenchbsetup working for compatibility.

```
\let\frenchbsetup\frenchsetup
\@onlypreamble\frenchbsetup
```

The following commands, defined with property .code in DeclareKeys{}, execute some post-treatment required to immediately take the flags value into account. The code is executed *only if* the corresponding option is *explicily set* in \frenchsetup{}.

```
\newcommand*{\FBSuppressWarning@setup}[1]%
  {\csname FBSuppressWarning#1\endcsname
   \ifFBSuppressWarning
     \renewcommand{\FBWarning}[1]{}%
   \fi
  }
\newcommand*{\FBStandardLayout@setup}[1]%
  {\ifFB@mainlanguage@FR
     \csname FBStandardLayout#1\endcsname
   \else
     \FBWarning{Option `StandardLayout' skipped:\MessageBreak
                French is *not* babel's last option.\MessageBreak
                Reported}%
   \fi
   \ifFBStandardLayout
     \FBStandardListSpacingtrue
     \FBStandardItemizeEnvtrue
     \FBStandardItemLabelstrue
     \FBStandardEnumerateEnvtrue
     \FBIndentFirstfalse
     \FBFrenchFootnotesfalse
     \FBAutoSpaceFootnotesfalse
   \else
     \FBStandardListSpacingfalse
     \FBStandardItemizeEnvfalse
     \FBStandardItemLabelsfalse
     \FBStandardEnumerateEnvfalse
     \FBIndentFirsttrue
     \FBFrenchFootnotestrue
```

```
    \FBAutoSpaceFootnotestrue
   \fi
  }
\newcommand*{\FBNosepItemize@setup}[1]%
  {\csname FBNosepItemize#1\endcsname
  }
\newcommand*{\FBNosepEnumerate@setup}[1]%
  {\csname FBNosepEnumerate#1\endcsname
  }
\newcommand*{\FBStandardLists@setup}[1]%
  {\csname FBStandardLists#1\endcsname
   \ifFBStandardLists
     \FBStandardListSpacingtrue
     \FBStandardItemizeEnvtrue
     \FBStandardEnumerateEnvtrue
     \FBStandardItemLabelstrue
   \else
     \FBStandardListSpacingfalse
     \FBStandardItemizeEnvfalse
     \FBStandardEnumerateEnvfalse
     \FBStandardItemLabelsfalse
   \fi
  }
\newcommand*{\FBThinColonSpace@setup}[1]%
  {\csname FBThinColonSpace#1\endcsname
   \ifFBThinColonSpace
     \renewcommand*{\FBcolonspace}{\FBthinspace}%
   \fi
  }
\newcommand*{\FBSmallCapsFigTabCaptions@setup}[1]%
  {\csname FBSmallCapsFigTabCaptions#1\endcsname
   \ifFBSmallCapsFigTabCaptions
   \else
     \let\FBfigtabshape\relax
   \fi
  }
\newcommand*{\FBCustomiseFigTabCaptions@setup}[1]%
  {\csname FBCustomiseFigTabCaptions#1\endcsname
   \ifFBCustomiseFigTabCaptions
     \FBWarning{Option CustomiseFigTabCaptions is *OBSOLETE*\MessageBreak
              The "caption" package is your friend,\MessageBreak
              see "frenchb.pdf" for more hints.\MessageBreak
              Trying to use endash though…\MessageBreak Reported}%
     \IfClassLoadedTF{memoir}%
       {\captiondelim{\space\textendash\space}}%
       {\ifFB@koma
```

```
          \renewcommand{\captionformat}{\space\textendash\space}%
        \else
          \IfClassLoadedTF{beamer}%
            {\setbeamertemplate{caption label separator}[endash]}%
            {\RequirePackage[labelsep=endash]{caption}}%
        \fi
       }%
   \fi
  }
\newcommand*{\FBINGuillSpace@setup}[1]%
  {\csname FBINGuillSpace#1\endcsname
   \ifFBINGuillSpace
     \FBsetspaces{guill}{1}{1}{1}%
   \fi
  }
\newcommand*{\FBEveryParGuill@setup}[1]%
  {\expandafter\let\expandafter
      \FBeveryparguill\csname FBguill#1\endcsname
   \ifx\FBeveryparguill\FBguillopen
   \else\ifx\FBeveryparguill\FBguillclose
        \else\ifx\FBeveryparguill\FBguillnone
             \else
               \let\FBeveryparguill\FBguillopen
               \FBWarning{Wrong value for `EveryParGuill':
                          try `open',\MessageBreak
                          `close' or `none'. Reported}%
             \fi
        \fi
   \fi
  }
\newcommand*{\FBEveryLineGuill@setup}[1]%
    {\expandafter\let\expandafter
        \FBeverylineguill\csname FBguill#1\endcsname
     \ifx\FBeverylineguill\FBguillopen
     \else\ifx\FBeverylineguill\FBguillclose
          \else\ifx\FBeverylineguill\FBguillnone
               \else
                 \let\FBeverylineguill\FBguillnone
                 \FBWarning{Wrong value for `EveryLineGuill':
                            try `open',\MessageBreak
                            `close' or `none'. Reported}%
               \fi
          \fi
     \fi
  }
```

This option has been kept for backward compatibility but is no longer necessary as

the \FB@addGUILspace attribute for LuaTeX is set to one (true) by default. A warning is issued.

```
\newcommand*{\FBog@setup}[1]{%
  \FBWarning{Options og=«, fg=» are not needed with LuaTeX.%
    \MessageBreak Automatic spacing of « » and ‹ › is active.%
    \MessageBreak Use \protect\NoAutoSpacing\space (inside a group) to%
    \MessageBreak cancel spacing locally.\MessageBreak Reported}
}
\newcommand*{\FBfg@setup}[1]{}
```

\FBprocess@options

\FBprocess@options will be executed just before \begin{document}: it first checks about packages loaded in the preamble (possibly after Babel) which customise lists: currently enumitem, paralist and enumerate; then it processes the options as set by \frenchsetup{} or forced for compatibility with packages loaded in the preamble.

When French is the main language, \extrasfrench and \captionsfrench are executed by Babel at \begin{document}, i.e. after \FBprocess@options.

```
\newcommand*{\FBprocess@options}{%
```

Only for legacy lists: update flags (and inform) if a package customising lists has been loaded, currently: enumitem, paralist, enumerate.

```
  \ifFBnewlists
  \else
    \IfPackageLoadedTF{enumitem}{%
      \ifFBStandardItemizeEnv
      \else
        \FBStandardItemizeEnvtrue
        \PackageInfo{french.ldf}%
          {Setting StandardItemizeEnv=true for\MessageBreak
           compatibility with enumitem package,\MessageBreak
           reported}%
       \fi
       \ifFBStandardEnumerateEnv
       \else
         \FBStandardEnumerateEnvtrue
         \PackageInfo{french.ldf}%
           {Setting StandardEnumerateEnv=true for\MessageBreak
            compatibility with enumitem package,\MessageBreak
            reported}%
       \fi}{}%
    \IfPackageLoadedTF{paralist}{%
      \ifFBStandardItemizeEnv
      \else
        \FBStandardItemizeEnvtrue
        \PackageInfo{french.ldf}%
          {Setting StandardItemizeEnv=true for\MessageBreak
```

```
          compatibility with paralist package,\MessageBreak
          reported}%
    \fi
    \ifFBStandardEnumerateEnv
    \else
      \FBStandardEnumerateEnvtrue
      \PackageInfo{french.ldf}%
        {Setting StandardEnumerateEnv=true for\MessageBreak
         compatibility with paralist package,\MessageBreak
         reported}%
    \fi}{}%
  \IfPackageLoadedTF{enumerate}{%
    \ifFBStandardEnumerateEnv
    \else
      \FBStandardEnumerateEnvtrue
      \PackageInfo{french.ldf}%
        {Setting StandardEnumerateEnv=true for\MessageBreak
         compatibility with enumerate package,\MessageBreak
         reported}%
    \fi}{}%
\fi
```

Options FrenchFootnotes and Option AutoSpaceFootnotes are handled now when new footnotes templates are available.

```
\ifFBnewfootnotes
  \FB@newFootnotesSetup
\fi
```

Option SmallCapsFigTabCaptions: \figurename and \tablename are printed in small caps (in French *only*), unless either SmallCapsFigTabCaptions is set to false or a class or package loaded defines \FBfigtabshape as \relax. As \figurename and \tablename should not include font commmands, we customise \fnum@figure and \fnum@table when available (not in beamer.cls f.i.).

```
\ifx\FBfigtabshape\relax
\else
  \ifdefined\fnum@figure
    \let\fnum@figureORI\fnum@figure
    \renewcommand{\fnum@figure}{{\ifFBfrench\FBfigtabshape\fi
                                 \fnum@figureORI}}%
  \fi
  \ifdefined\fnum@table
    \let\fnum@tableORI\fnum@table
    \renewcommand{\fnum@table}{{\ifFBfrench\FBfigtabshape\fi
                                \fnum@tableORI}}%
  \fi
\fi
```

AutoSpacePunctuation, when true, adds a non-breaking space (in French only) before the four characters (:;!?) if and only if spacing is required by French typographic rules. When false, these characters are left unchanged.

```
\ifFBAutoSpacePunctuation
  \autospace@beforeFDP
\else
  \noautospace@beforeFDP
  \FBWarning{AutoSpacePunctuation should *not* be set to false%
  \MessageBreak in LuaTeX, unless you know what you are doing.%
  \MessageBreak Reported}
\fi
```

When OriginalTypewriter is set to false (the default), a *hook* is added to \ttfamily, to prevent addition of automatic spaces before the four active characters in computer code. \rmfamily and \sffamily need to be redefined also, as \ttfamily is not always used inside a group.

```
\ifFBOriginalTypewriter
\else
  \AddToHook{ttfamily}{\FB@spacing=\z@}%
  \AddToHook{rmfamily}{\FB@spacing=\@ne}%
  \AddToHook{sffamily}{\FB@spacing=\@ne}%
\fi
```

When package numprint is loaded with option autolanguage, numprint's command \npstylefrench has to be redefined differently according to the value of flag ThinSpaceInFrenchNumbers. As \npstylefrench was undefined in old versions of numprint, we provide this command.

```
\IfPackageLoadedTF{numprint}%
  {\ifnprt@autolanguage
     \providecommand*{\npstylefrench}{}%
     \ifFBThinSpaceInFrenchNumbers
       \renewcommand*{\FBthousandsep}{\FBthinspace}%
     \fi
     \g@addto@macro\npstylefrench{\npthousandsep{\FBthousandsep}}%
   \fi
  }{}%
```

FrenchSuperscripts: if true, try to take advantage of the realscripts package if it has been loaded. In case the current font has no real superscripts (lmodern…), \fup is preferred to \fakesuperscript. The star-form \up*=\fup is provided for fonts that lack some superior letters: f.i. Adobe Jenson Pro has no superiors for "c,f,g,j,k,p,q".

```
\ifFBFrenchSuperscripts
  \IfPackageLoadedTF{realscripts}%
    {\RenewDocumentCommand\fakesuperscript{m}{\fup{##1}}%
     \NewDocumentCommand\FB@up{m}{%
```

```
            \realsuperscript{\FB@lc{##1}}}%
        \DeclareRobustCommand*{\up}{%
            \texorpdfstring{\@ifstar{\fup}{\FB@up}}%
                           {}%
            }%
        }
        {\DeclareRobustCommand*{\up}{%
            \texorpdfstring{\@ifstar{\fup}{\fup}}%
                           {}%
            }%
        }
\else
```

If `false`, use the standard command `\textsuperscript`. The star-form `\up*` remains defined as `\fup`. When `realscripts` has been loaded, `\textsuperscript` is `\realsuperscript`, uppercased argument would be printed as is (most fonts do not have superscripts for uppercased letters).

```
    \NewDocumentCommand\FB@up{m}{%
        \textsuperscript{\FB@lc{##1}}}%
    \DeclareRobustCommand*{\up}{%
        \texorpdfstring{\@ifstar{\fup}{\FB@up}}%
                       {}%
        }%
\fi
```

`LowercaseSuperscripts`: if `false` `\FB@lc` is redefined to do nothing.

```
\ifFBLowercaseSuperscripts
\else
  \renewcommand*{\FB@lc}[1]{##1}%
\fi
```

Option `UnicodeNoBreakSpaces` is meant for HTML translators: when true, all non-breaking spaces added by `babel-french` are coded in the PDF file as Unicode characters, namely U+A0 or U+202F, instead of penalties and glues.

```
\ifFBUnicodeNoBreakSpaces
  \FB@ucsNBSP=\@ne
  \renewcommand*{\FBmedkern}{\char"202F\relax}%
  \renewcommand*{\FBthickkern}{\char"A0\relax}%
  \ifFBThinSpaceInFrenchNumbers
    \renewcommand*{\FBthousandsep}{\char"202F\relax}%
  \else
    \renewcommand*{\FBthousandsep}{\char"A0\relax}%
  \fi
\fi
```

TocPartNameFull: for memoir and koma-script classes only. \KOMAoptions cannot be changed 'AtBeginDocument', executing \FBprocess@options just before is fine.

```
\ifFB@koma
   \ifFBTocPartNameFull
     \KOMAoptions{toc=flat, numbers=nodotatend}%
     \renewcommand*{\addparttocentry}[2]{%
        \addtocentrydefault{part}{\FBtocpartname{##1}}{##2}}%
   \fi
\fi
\IfClassLoadedTF{memoir}%
   {\ifFBTocPartNameFull
      \renewcommand{\partnumberline}[1]{\FBtocpartname{##1}}%
    \fi
   }{}%
```

ShowOptions: if true, print the list of all options to the .log file.

```
\ifFBShowOptions
  \GenericWarning{* }{%
   *** List of possible options for babel-french ***\MessageBreak
   [Default values between brackets when french is loaded *LAST*]%
   \MessageBreak
   ShowOptions [false]\MessageBreak
   StandardLayout [false]\MessageBreak
   PartNameFull [true]\MessageBreak
   TocPartNameFull [true]\MessageBreak
   IndentFirst [true]\MessageBreak
   ListItemsAsPar [false]\MessageBreak
   StandardListSpacing [false]\MessageBreak
   NosepItemize [true]\MessageBreak
   NosepEnumerate [false]\MessageBreak
   StandardItemLabels [false]\MessageBreak
   ItemLabels=\textemdash, \textbullet,
      \protect\ding{43},... [\textendash]\MessageBreak
   ItemLabeli=\textemdash, \textbullet,
      \protect\ding{43},... [\textendash]\MessageBreak
   ItemLabelii=\textemdash, \textbullet,
      \protect\ding{43},... [\textendash]\MessageBreak
   ItemLabeliii=\textemdash, \textbullet,
      \protect\ding{43},... [\textendash]\MessageBreak
   ItemLabeliv=\textemdash, \textbullet,
      \protect\ding{43},... [\textendash]\MessageBreak
   StandardLists [false]\MessageBreak
   FrenchFootnotes [true]\MessageBreak
   AutoSpaceFootnotes [true]\MessageBreak
   AutoSpacePunctuation [true]\MessageBreak
   ThinColonSpace [false]\MessageBreak
```

```
    ThinSpaceInFrenchNumbers [false]\MessageBreak
    UnicodeNoBreakSpaces [false]\MessageBreak
    OriginalTypewriter [false]\MessageBreak
    INGuillSpace [false]\MessageBreak
    EveryParGuill=open, close, none [open]\MessageBreak
    EveryLineGuill=open, close, none
                  [open in LuaTeX, none otherwise]\MessageBreak
    InnerGuillSingle [false]\MessageBreak
    SmallCapsFigTabCaptions [true]\MessageBreak
    FrenchSuperscripts [true]\MessageBreak
    LowercaseSuperscripts [true]\MessageBreak
    SuppressWarning [false]\MessageBreak
    \MessageBreak
    ****************************************%
    \MessageBreak\protect\frenchsetup{ShowOptions}}
  \fi
}
```

Just before \begin{document}, let's now process the remaining options, either not explicitly set by \frenchsetup{} or possibly modified by packages loaded after babel-french. We also have to provide an \xspace command in case the xspace package is not loaded. In some cases (package standalone, dtk,…) several \documentclass{} commands are allowed, so use \AddToHookNext instead of \AddToHook (all instances) as our stuff should only added to the first occurence of \documentclass{} anyway.

```
\AddToHookNext{env/document/before}{%
  \providecommand*{\xspace}{\relax}%
  \FBprocess@options
}
```

2.10 French lists

2.10.1 Code shared by new and legacy lists

\listindentFB
\descindentFB
\labelwidthFB
\labelfullwidthFB

Let's define three dimens \listindentFB, \descindentFB, and \labelwidthFB to customise lists' horizontal indentations. They are given silly negative values here in order to eventually enable their customisation in the preamble. They will get reasonnable defaults later when entering French (see below \setlistindentFB and \setlabelitemsFB) unless they have been customised before.

```
\newdimen\listindentFB
\setlength{\listindentFB}{-1pt}
\newdimen\descindentFB
\setlength{\descindentFB}{-1pt}
\newdimen\labelwidthFB
\setlength{\labelwidthFB}{-1pt}
\newdimen\labelfullwidthFB
```

The next function will be included in \setup@FBnewlists or \setup@FBlegacylists which are exectuted in \extrasfrench{} 'AtBeginDocument'.

```
\def\setlistindentFB{%
  \ifdim\listindentFB<\z@
    \ifdim\parindent=\z@
      \setlength{\listindentFB}{1.5em}%
    \else
      \setlength{\listindentFB}{\parindent}%
    \fi
  \fi
  \ifdim\descindentFB<\z@
    \setlength{\descindentFB}{\listindentFB}%
  \fi
}
```

Let's consider French itemize-lists. They differ from those provided by the standard LaTeX classes:

- The '•' is never used in French itemize-lists, an emdash '—' or an endash '–' is preferred for all levels. The item label to be used in French, stored in \FrenchLabelItem}, defaults to '—' and can be changed using \frenchsetup{} (see section 2.9).
- Vertical spacing between items, before and after the list, should be *null* with *no glue* added;
- The labels of itemize-lists are vertically aligned as shown p. 6.

\FrenchLabelItem
\Frlabelitemi
\Frlabelitemii
\Frlabelitemiii
\Frlabelitemiv

Default labels for French itemize-lists —same label for all levels—, (already defined as empty by \DeclareKey{}):

```
\renewcommand*{\FrenchLabelItem}{\textemdash}
\renewcommand*{\Frlabelitemi}{\FrenchLabelItem}
\renewcommand*{\Frlabelitemii}{\FrenchLabelItem}
\renewcommand*{\Frlabelitemiii}{\FrenchLabelItem}
\renewcommand*{\Frlabelitemiv}{\FrenchLabelItem}
```

The next function will be included in \setup@FBnewlists or \setup@FBlegacylists which are exectuted in \extrasfrench{} 'AtBeginDocument'.

```
\def\setlabelitemsFB{%
  \let\labelitemi\Frlabelitemi
  \let\labelitemii\Frlabelitemii
  \let\labelitemiii\Frlabelitemiii
  \let\labelitemiv\Frlabelitemiv
}
```

2.10.2 Code for legacy lists only

\listFB
\listORI
\FB@listVsettings

Vertical spacing in lists should be shorter in French texts than the defaults provided by LaTeX. Note that the easy way, just changing values of vertical spacing parameters when entering French and restoring them to their defaults on exit would not work; so we define the command \FB@listVsettings to hold the settings to be used by the French variant \listFB of \list. Note that switching to \listFB reduces vertical spacing in *all* environments built on \list: itemize, enumerate, description, but also abstract, quotation, quote and verse…
The amount of vertical space before and after a list is given by \topsep + \parskip (+ \partopsep if the list starts a new paragraph). IMHO, \parskip should be added *only* when the list starts a new paragraph, so I subtract \parskip from \topsep and add it back to \partopsep; this will normally make no difference because \parskip's default value is 0pt, but will be noticeable when \parskip is *not* null.

```
\ifFBnewlists
\else
  \let\listORI\list
  \let\endlistORI\endlist
  \newdimen\FB@pardim
  \def\FB@listVsettings{%
      \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
      \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
```

\parskip is of type 'skip', its mean value only (*not the glue*) should be subtracted from \topsep and added to \partopsep, so convert \parskip to a 'dimen' using \FB@pardim.

```
      \FB@pardim=\parskip
```

If \parskip is not null, \parsep is set to \parskip, so paragraphs inside items will be preceeded by the same vertical space as paragraphs located outside lists; the vertical skip before items (\itemsep + \parsep) doesn't need to be enlarged.

```
      \ifdim\FB@pardim>\z@
        \addtolength{\topsep}{-\FB@pardim}%
        \addtolength{\partopsep}{\FB@pardim}%
        \setlength{\parsep}{\FB@pardim}%
        \addtolength{\itemsep}{-\FB@pardim}%
      \fi
  }
  \def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
  \let\endlistFB\endlistORI
```

\FB@listHsettings

\FB@listHsettings holds the new horizontal settings chosen for French lists itemize, enumerate and description (two possible layouts).

```
\def\FB@listHsettings{%
  \ifdim\labelwidthFB<\z@
    \settowidth{\labelwidthFB}{\FrenchLabelItem}%
  \fi
  \labelfullwidthFB=\labelwidthFB
  \advance\labelfullwidthFB by \labelsep
  \ifFBListItemsAsPar
```

Optional layout: lists' items are typeset as paragraphs with indented labels.

```
    \leftmargini=\z@
    \itemindent=\labelfullwidthFB
    \advance\itemindent by \listindentFB
    \bbl@for\FB@dp {2, 3, 4, 5, 6}%
      {\csname leftmargin\romannumeral\FB@dp\endcsname =
         \listindentFB}%
  \else
```

Default layout: labels hanging into the list left margin.

```
    \bbl@for\FB@dp {1, 2, 3, 4, 5, 6}%
      {\csname leftmargin\romannumeral\FB@dp\endcsname =
         \labelfullwidthFB}%
    \advance\leftmargini by \listindentFB
    \itemindent=\z@
```

Same 'parindent' for paragraphs in lists' items (was null as in standard lists).

```
  \fi
  \listparindent=\parindent
  \leftmargin=\csname leftmargin%
     \ifnum\@listdepth=\@ne i\else ii\fi\endcsname
}
```

`\itemizeFB` New environment for French itemize-lists.
`\FB@itemizeVsettings` `\FB@itemizeVsettings` suppresses all vertical spaces including glue unless option `StandardListSpacing` is set.

```
\def\FB@itemizeVsettings{%
  \ifFBStandardListSpacing
  \else
    \FB@pardim=\parskip
    \ifdim\FB@pardim>\z@
      \setlength{\topsep}{-\FB@pardim}%
      \setlength{\partopsep}{\FB@pardim}%
      \setlength{\parsep}{\FB@pardim}%
      \setlength{\itemsep}{-\FB@pardim}%
    \else
      \setlength{\topsep}{\z@}%
      \setlength{\partopsep}{\z@}%
```

```
        \setlength{\parsep}{\z@}%
        \setlength{\itemsep}{\z@}%
      \fi
    \fi
}
```

The definition of \itemizeFB follows the one of \itemize in standard LaTeX classes (see ltlists.dtx), vertical spaces are customised by \FB@itemizeVsettings.

```
\def\itemizeFB{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth by \@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \expandafter
    \listORI
    \csname\@itemitem\endcsname
    {\FB@itemizeVsettings
     \settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
     \FB@listHsettings
    }%
  \fi
}
\let\enditemizeFB\endlistORI
```

\enumerateFB The definition of \enumerateFB, new to version 2.6a, follows the one of \enumerate in standard LaTeX classes (see ltlists.dtx), vertical spaces are customised (or not) via \list (=\listFB or \listORI) and horizontal spaces (leftmargins) are borrowed from itemize lists via \FB@listHsettings.

```
\def\enumerateFB{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth by \@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
    \expandafter
    \list
      \csname label\@enumctr\endcsname
      {\FB@listHsettings
       \usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi
}
\let\endenumerateFB\endlistORI
```

\descriptionFB Same tuning for the description environment (see classes.dtx for the original definition). Customisable dimen \descindentFB, which defaults to \listindentFB, is added to \itemindent (first level only). When \descindentFB=0pt (1rst level labels start at the left margin), \leftmargini is reduced to \listindentFB instead of \listindentFB + \labelfullwidthFB.

When option ListItemsAsPar is turned to true, the description items are also displayed as paragraphs; \descindentFB=0pt can be used to push labels to the left margin.

```
\def\descriptionFB{%
  \list{}{\FB@listHsettings
          \labelwidth=\z@
          \ifFBListItemsAsPar
            \itemindent=\descindentFB
          \else
            \itemindent=-\leftmargin
            \ifnum\@listdepth=\@ne
              \ifdim\descindentFB=\z@
                \ifdim\listindentFB>\z@
                  \leftmargini=\listindentFB
                  \leftmargin=\leftmargini
                  \itemindent=-\leftmargin
                \fi
              \else
                \advance\itemindent by \descindentFB
              \fi
            \fi
          \fi
          \let\makelabel\descriptionlabel
         }%
\}
\let\enddescriptionFB\endlistORI
\fi
```

\setup@FBlegacylists This command gathers the customisation of French lists when the new templates are not available: the legacy lists are redefined.

```
\def\setup@FBlegacylists{%
   \setlistindentFB
   \ifFBStandardListSpacing
   \else \let\list\listFB \fi
   \ifFBStandardItemizeEnv
   \else \let\itemize\itemizeFB \fi
   \ifFBStandardItemLabels
   \else \setlabelitemsFB \fi
   \ifFBStandardEnumerateEnv
   \else \let\enumerate\enumerateFB \let\description\descriptionFB \fi
\}
\def\warning@FBlegacylists{%
  \FBWarning{You are relying on the legacy list code.\MessageBreak
        There is nothing wrong with this, just be aware\MessageBreak
        that new lists templates are available and will\MessageBreak
        become the default sooner or later.\MessageBreak
```

```
      Adding \protect\DocumentMetadata{ ... } before the\MessageBreak
    \protect\documentclass{} command, enables the new\MessageBreak
      lists templates.  Give them a try!\MessageBreak Reported}%
}
```

2.10.3 Code for new lists only

\setup@FBnewlists Customization possibilities for the new lists templates:

```
\ifFBnewlists
  \newdimen\itemindentFB
  \newdimen\leftmarginiFB
  \newdimen\leftmarginiiFB
  \def\setup@FBnewlists{%
    \setlistindentFB
```

Compute \leftmarginiFB, \leftmarginiiFB and \itemidentFB for both possible layouts (default and ListItemsAsPar).

```
    \ifdim\labelwidthFB<\z@
      \settowidth{\labelwidthFB}{\FrenchLabelItem}%
    \fi
    \labelfullwidthFB=\labelwidthFB
    \advance\labelfullwidthFB by \labelsep
    \ifFBListItemsAsPar
      \leftmarginiFB=\z@
      \leftmarginiiFB=\listindentFB
      \itemindentFB=\labelfullwidthFB
      \advance\itemindentFB by \listindentFB
    \else
      \leftmarginiFB=\listindentFB
      \advance\leftmarginiFB by \labelfullwidthFB
      \leftmarginiiFB=\labelfullwidthFB
      \itemindentFB=\z@
    \fi
    \listparindent=\parindent
```

Let's define lists' variants with tight or nosep vertical spacing (6 levels).

```
    \DeclareInstance{block}{tight-list-1}{std}{%
         begin-vspace = 0.8ex plus 0.4ex minus 0.4ex ,
         begin-extra-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
         para-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
         item-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
         left-margin:v = leftmarginiFB ,
    }%
    \DeclareInstance{block}{tight-list-2}{std}{%
         begin-vspace = 0.8ex plus 0.4ex minus 0.4ex ,
         begin-extra-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
```

```
      para-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
      item-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
      left-margin:v = leftmarginiiFB ,
  }%
  \DeclareInstanceCopy{block}{tight-list-3}{tight-list-2}%
  \DeclareInstanceCopy{block}{tight-list-4}{tight-list-2}%
  \DeclareInstanceCopy{block}{tight-list-5}{tight-list-2}%
  \DeclareInstanceCopy{block}{tight-list-6}{tight-list-2}%
  %
  \DeclareInstance{block}{nosep-list-1}{std}{%
      begin-vspace = 0pt ,
      begin-extra-vspace = 0pt ,
      para-vspace = 0pt ,
      item-vspace = 0pt ,
      left-margin:v = leftmarginiFB ,
  }%
  \DeclareInstance{block}{nosep-list-2}{std}{%
      begin-vspace = 0pt ,
      begin-extra-vspace = 0pt ,
      para-vspace = 0pt ,
      item-vspace = 0pt ,
      left-margin:v = leftmarginiiFB ,
  }%
  \DeclareInstanceCopy{block}{nosep-list-3}{nosep-list-2}%
  \DeclareInstanceCopy{block}{nosep-list-4}{nosep-list-2}%
  \DeclareInstanceCopy{block}{nosep-list-5}{nosep-list-2}%
  \DeclareInstanceCopy{block}{nosep-list-6}{nosep-list-2}%
```

Itemize lists vertical spacing (std, tight or nosep):

```
  \ifFBStandardItemizeEnv
  \else
    \ifFBStandardListSpacing
    \else
      \ifFBNosepItemize
        \EditInstance{blockenv}{itemize}{%
           block-instance = nosep-list ,
        }%
      \else
        \EditInstance{blockenv}{itemize}{%
           block-instance = tight-list ,
        }%
      \fi
    \fi
    \EditInstance{list}{itemize-1}{%
       item-indent:v = itemindentFB ,
       label-width:v = labelfullwidthFB ,
    }%
```

```
    \EditInstance{list}{itemize-2}{%
       item-indent:v = itemindentFB ,
       label-width:v = labelfullwidthFB ,
    }%
    \EditInstance{list}{itemize-3}{%
       item-indent:v = itemindentFB ,
       label-width:v = labelfullwidthFB ,
    }%
    \EditInstance{list}{itemize-4}{%
       item-indent:v = itemindentFB ,
       label-width:v = labelfullwidthFB ,
    }%
  \fi
```

Labels for Itemize lists (4 levels):

```
  \ifFBStandardItemLabels
  \else
    \setlabelitemsFB
  \fi
```

Enumerate and description lists vertical spacing (std, tight or nosep):

```
  \ifFBStandardEnumerateEnv
  \else
    \ifFBStandardListSpacing
    \else
      \ifFBNosepEnumerate
        \EditInstance{blockenv}{enumerate}{%
           block-instance = nosep-list ,
        }%
        \EditInstance{blockenv}{description}{%
           block-instance = nosep-list ,
           inner-instance = description ,
        }%
      \else
        \EditInstance{blockenv}{enumerate}{%
           block-instance = tight-list ,
        }%
        \EditInstance{blockenv}{description}{%
           block-instance = tight-list ,
           inner-instance = description ,
        }%
      \fi
    \fi
```

Enumerate and description lists indentation (4 levels for enumerate, 6 levels for description):

```
 \EditInstance{list}{enumerate-1}{%
    item-indent:v = itemindentFB ,
    label-width:v = labelfullwidthFB ,
 }%
 \EditInstance{list}{enumerate-2}{%
    item-indent:v = itemindentFB ,
    label-width:v = labelfullwidthFB ,
 }%
 \EditInstance{list}{enumerate-3}{%
    item-indent:v = itemindentFB ,
    label-width:v = labelfullwidthFB ,
 }%
 \EditInstance{list}{enumerate-4}{%
    item-indent:v = itemindentFB ,
    label-width:v = labelfullwidthFB ,
 }%
 %
\ifFBListItemsAsPar
  \leftmarginiFB=\z@
  \itemindentFB=\descindentFB
\else
  \leftmarginiFB=\descindentFB
  \advance\leftmarginiFB by \labelfullwidthFB
  \itemindentFB=-\labelwidthFB
\fi
\EditInstance{list}{description-1}{%
    item-indent:v = itemindentFB ,
    label-width = 0pt ,
 }%
\EditInstance{list}{description-2}{%
    item-indent:v = itemindentFB ,
    label-width = 0pt ,
 }%
\EditInstance{list}{description-3}{%
    item-indent:v = itemindentFB ,
    label-width = 0pt ,
 }%
\EditInstance{list}{description-4}{%
    item-indent:v = itemindentFB ,
    label-width = 0pt ,
 }%
\EditInstance{list}{description-5}{%
    item-indent:v = itemindentFB ,
    label-width = 0pt ,
 }%
\EditInstance{list}{description-6}{%
```

```
          item-indent:v = itemindentFB ,
          label-width = 0pt ,
       }%
    \fi
```

Quote (and verse) and quotation (and abstract) environments vertical spacing (standard or tight):

```
    \ifFBStandardListSpacing
    \else
      \EditInstance{block}{quote-1}{%
         begin-vspace = 0.8ex plus 0.4ex minus 0.4ex ,
         begin-extra-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
         para-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
      }%
      \DeclareInstanceCopy{block}{quote-2}{quote-1}%
      \DeclareInstanceCopy{block}{quote-3}{quote-1}%
      \DeclareInstanceCopy{block}{quote-4}{quote-1}%
      \DeclareInstanceCopy{block}{quote-5}{quote-1}%
      \DeclareInstanceCopy{block}{quote-6}{quote-1}%
      \EditInstance{block}{quotation-1}{%
         begin-vspace = 0.8ex plus 0.4ex minus 0.4ex ,
         begin-extra-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
         para-vspace = 0.4ex plus 0.2ex minus 0.2ex ,
      }%
      \DeclareInstanceCopy{block}{quotation-2}{quotation-1}%
      \DeclareInstanceCopy{block}{quotation-3}{quotation-1}%
      \DeclareInstanceCopy{block}{quotation-4}{quotation-1}%
      \DeclareInstanceCopy{block}{quotation-5}{quotation-1}%
      \DeclareInstanceCopy{block}{quotation-6}{quotation-1}%
    \fi
   }
\fi
```

`\bbl@frenchlistlayout` Nothing has to be done at language's switches regarding lists, except at the first switch in case French is the main language, then lists are set up once for all. There is nothing to do for lists in `\noextrasfrench`.

```
\def\bbl@frenchlistlayout{%
  \ifFB@mainlanguage@FR
    \ifFBnewlists
      \setup@FBnewlists
      \let\setup@FBnewlists\relax
    \else
```

Warnings: no list customisation with new templates if LaTeX Format < 2026-06-01.

```
      \IfDocumentMetadataTF
        {\warning@FBLaTeXFormat
```

```
          \global\let\warning@FBLaTeXFormat\relax
        }%
        {\warning@FBlegacylists
         \global\let\warning@FBlegacylists\relax
         \setup@FBlegacylists
         \let\setup@FBlegacylists\relax
        }%
    \fi
  \fi}
\addto\extrasfrench{\bbl@frenchlistlayout}
```

2.11 French indentation of sections

\bbl@frenchindent

In French the first paragraph of each section should be indented, this is another difference with US-English. This is controlled by the flag `\if@afterindent`.

Indentation changes at language switches only if `IndentFirst=true` and French isn't the main language.

```
\def\bbl@frenchindent{%
  \ifFBIndentFirst
    \ifFB@mainlanguage@FR\else\babel@save\@afterindentfalse\fi
    \let\@afterindentfalse\@afterindenttrue
    \@afterindenttrue
  \fi}
\addto\extrasfrench{\bbl@frenchindent}
```

2.12 Formatting footnotes

The layout of footnotes is controlled by two flags `\ifFBAutoSpaceFootnotes` and `\ifFBFrenchFootnotes` which are set by options of `\frenchsetup{}` (see section 2.9). The layout of footnotes only depends on the main language (French or other). Two cases: new templates or legacy code.

2.13 Common settings for both new and old footnote's code

\parindentFFN
\dotFFN
\kernFFN

The value of `\parindentFFN` will be redefined at the `\begin{document}`, as the maximum of `\parindent` and 1.8em (like `\footnotemargin`) *unless* it has been set in the preamble.

```
\newdimen\parindentFFN
\parindentFFN=\maxdimen
```

In French, the number preceeding the footnote text is typeset in normal size (not superscript) followed by a dot and a space, both are customisable.

```
\newcommand*{\dotFFN}{.}
\newcommand*{\kernFFN}{\kern .5em}
```

\FBfnmarkspace Let's define a customisable thin space which will be added before footnote's call.

```
\newcommand*{\FBfnmarkspace}{\kern .5\fontdimen2\font}
```

2.13.1 Code for the new footnotes templates

\FB@newFootnotesSetup \FB@newFootnotesSetup will be processed by \FBprocess@options just before \begin{document}.

```
\ifFBnewfootnotes
  \providecommand*{\multiplefootnotemarker}{3sp}
  \newcommand*{\FBfnmark}{%
    \ifdim\lastkern=\multiplefootnotemarker
    \else \FBfnmarkspace \fi}
  \newcommand*{\FB@newFootnotesSetup}{%
    \ifdim\parindentFFN<\maxdimen
    \else
       \parindentFFN=\parindent
       \ifdim\parindentFFN<1.8em \parindentFFN=1.8em \fi
    \fi
    \ifFBFrenchFootnotes
      \NewSocketPlug{fntext/mark}{FBfnmark}
        {\hb@xt@ \parindentFFN{\hss\@thefnmark}\dotFFN\kernFFN}
      \AssignSocketPlug{fntext/mark}{FBfnmark}
      \AddToHook{cmd/maketitle/before}[.]
         {\AssignSocketPlug{fntext/mark}{default}}
      \AddToHook{cmd/maketitle/after}[.]
         {\AssignSocketPlug{fntext/mark}{FBfnmark}}
      \AddToHook{env/minipage/begin}[.]
         {\AssignSocketPlug{fntext/mark}{default}}
      \AddToHook{fntext/para}[.]{\parindent=\parindentFFN}
      \AddToHook{fntext/para}[.]{\localleftbox{}}
      \AddToHook{fntext/para}[.]{\let\FBeverypar@quote\relax}
    \fi
    \ifFBAutoSpaceFootnotes
      \AddToHook{fnmark/before}[.]{\FBfnmark}
    \fi
}
\fi
```

2.13.2 Code for legacy footnotes

\@makefntextFB We define \@makefntextFB, a variant of \@makefntext which is responsible for the layout of footnotes, to match the specifications of the French 'Imprimerie Nationale': footnotes will be indented by \parindentFFN, numbers (if any) typeset on the baseline (instead of superscripts), right aligned on \parindentFFN and followed by a dot and an half quad kern. Whenever symbols are used to number footnotes (as in \thanks

for instance), we switch back to the standard layout (the French layout of footnotes is meant for footnotes numbered by arabic or roman digits).

\@makefntextFB's definition depends on the document's class.

\FBfnindent will be set later on to the width of the box holding the footnote mark, \dotFFN and \kernFFN (flushed right). It is used by memoir and koma-script classes.

```
\ifFBnewfootnotes
\else
  \newdimen\FBfnindent
```

Koma-script classes: they provide \deffootnote, a handy command to customise the footnotes' layout (see English manual scrguien.pdf); it redefines \@makefntext and \@@makefnmark. First, save the original definitions.

```
  \ifFB@koma
    \let\@makefntextORI\@makefntext
    \let\@@makefnmarkORI\@@makefnmark
```

\@makefntextFB and \@@makefnmarkFB are used when option FrenchFootnotes is true.

```
    \deffootnote[\FBfnindent]{\z@}{\parindentFFN}%
                {\thefootnotemark\dotFFN\kernFFN}
    \let\@makefntextFB\@makefntext
    \let\@@makefnmarkFB\@@makefnmark
```

\@makefntextTH and \@@makefnmarkTH are meant for the \thanks command used by \maketitle when FrenchFootnotes is true.

```
    \deffootnote[\parindentFFN]{\z@}{\parindentFFN}%
                {\textsuperscript{\thefootnotemark}}
    \let\@makefntextTH\@makefntext
    \let\@@makefnmarkTH\@@makefnmark
```

Restore the original definitions.

```
    \let\@makefntext\@makefntextORI
    \let\@@makefnmark\@@makefnmarkORI
  \fi
```

Definitions for the memoir class:

```
  \IfClassLoadedTF{memoir}
```

(see original definition in memman.pdf)

```
    {\newcommand{\@makefntextFB}[1]{%
       \def\footscript##1{##1\dotFFN\kernFFN}%
       \setlength{\footmarkwidth}{\FBfnindent}%
       \setlength{\footmarksep}{-\footmarkwidth}%
       \setlength{\footparindent}{\parindentFFN}%
```

```
      \makefootmark #1}%
    }{}
```

Definitions for the beamer class:

the original definition is in beamerbaseframecomponents.sty, note that for the beamer class footnotes are LR-boxes, not paragraphs, so \parindentFFN is irrelevant.

```
\IfClassLoadedTF{beamer}
   {\def\@makefntextFB#1{%
      \def\insertfootnotetext{#1}%
      \def\insertfootnotemark{\insertfootnotemarkFB}%
      \usebeamertemplate***{footnote}}%
     \def\insertfootnotemarkFB{%
      \usebeamercolor[fg]{footnote mark}%
      \usebeamerfont*{footnote mark}%
      \llap{\@thefnmark}\dotFFN\kernFFN}%
    }{}
```

Now the default definition of \@makefntextFB for standard LaTeX and AMS classes. The next command prints the footnote mark according to the specifications of the French 'Imprimerie Nationale'. Keep in mind that \@thefnmark might be empty (i.e. in AMS classes' titles)!

```
\providecommand*{\insertfootnotemarkFB}{%
  \parindent=\parindentFFN
  \rule\z@\footnotesep
  \setbox\@tempboxa\hbox{\@thefnmark}%
  \ifdim\wd\@tempboxa>\z@
    \llap{\@thefnmark}\dotFFN\kernFFN
  \fi}
\providecommand\@makefntextFB[1]{\insertfootnotemarkFB #1}
```

The rest of \@makefntext's customisation will be done at the \begin{document}: saving the original definition of \@makefntext, then redefining \@makefntext according to the value of flag \ifFBFrenchFootnotes (true or false).

\@footnotemark We will save the original definition of \@footnotemark at \begin{document} in order to include any customisation that packages might have done; we define a variant \@footnotemarkFB which just adds a (customisable) thin space before the number or symbol calling a footnote (any space typed in is removed first). The choice between the two definitions (valid for the whole document) is controlled by flag \ifFBAutoSpaceFootnotes.

\@footnotemark's customisation: \FBfnmarkspace will be added before footnote's call by \@footnotemarkFB.

```
\def\@footnotemarkFB{\leavevmode\unskip\unkern
                     \protect\FBfnmarkspace\@footnotemarkORI}%
```

The following command \FB@legacyFootnoteSetup will be processed just before \begin{document} by \FBprocess@options . It gathers the customisation of footnotes in French.
The LuaTeX command \localleftbox and \FBeverypar@quote used by \frquote{} have to be reset inside footnotes; done for LaTeX based formats only.

```
\newcommand*{\FB@legacyFootnoteSetup}{%
```

When the footnotebackref package is loaded, babel-french will not customise \@footnotetext in order to keep back referencing working.

```
  \IfPackageLoadedTF{footnotebackref}%
    {\FBFrenchFootnotesfalse
     \FBWarning
        {footnotebackref package loaded.\MessageBreak
         babel-french will NOT customise footnotes;%
         \MessageBreak reported}}%
    {}%
```

The bigfoot package deeply changes the way footnotes are handled. When bigfoot is loaded, we just warn the user that babel-french will not customise footnotes at all.

```
  \IfPackageLoadedTF{bigfoot}%
    {\FBWarning
        {bigfoot package in use.\MessageBreak
         babel-french will NOT customise footnotes;%
         \MessageBreak reported}}%
```

Otherwise, footnotes may be customised according to the \frenchsetup{} options.

```
    {\let\@footnotemarkORI\@footnotemark
     \ifFBAutoSpaceFootnotes
       \let\@footnotemark\@footnotemarkFB
     \fi
     \ifdim\parindentFFN<\maxdimen
     \else
       \parindentFFN=\parindent
       \ifdim\parindentFFN<1.5em \parindentFFN=1.5em \fi
     \fi
     \settowidth{\FBfnindent}{\dotFFN\kernFFN}%
     \addtolength{\FBfnindent}{\parindentFFN}%
     \let\@makefntextORI\@makefntext
```

Koma-script classes require a special treatment.
Definition of \@makefntext for koma-script classes: running makefntextORI inside a group to reset \localleftbox{} and \FBeverypar@quote would mess up the layout of footnotes whenever the first manadatory argument of \deffootnote{} (used as \leftskip) is non-nil (default is 1em, 0pt in French).

```
     \ifFB@koma
       \let\@@makefnmarkORI\@@makefnmark
```

```
      \long\def\@makefntext##1{%
        \localleftbox{}%
        \let\FBeverypar@save\FBeverypar@quote
        \let\FBeverypar@quote\relax
        \ifFBFrenchFootnotes
          \ifx\footnote\thanks
            \let\@@makefnmark\@@makefnmarkTH
            \@makefntextTH{##1}
          \else
            \let\@@makefnmark\@@makefnmarkFB
            \@makefntextFB{##1}
          \fi
        \else
          \let\@@makefnmark\@@makefnmarkORI
          \@makefntextORI{##1}%
        \fi
        \let\FBeverypar@quote\FBeverypar@save
        \localleftbox{\FBeveryline@quote}}%
    \else
```

Special add-on for the `memoir` class: `\@makefntext` is redefined as `\makethanksmark` by `\maketitle`, hence these settings to match the other notes' vertical alignment.

```
      \IfClassLoadedTF{memoir}%
         {\ifFBFrenchFootnotes
            \setlength{\thanksmarkwidth}{\parindentFFN}%
            \setlength{\thanksmarksep}{-\thanksmarkwidth}%
          \fi
         }{}%
```

Special add-on for the `beamer` class: issue a warning in case `\parindentFFN` has been changed.

```
      \IfClassLoadedTF{beamer}%
         {\ifFBFrenchFootnotes
            \ifdim\parindentFFN=1.5em\else
              \FBWarning{%
                 \protect\parindentFFN\space is ineffective%
                 \MessageBreak within the beamer class.%
                 \MessageBreak Reported}%
            \fi
          \fi
         }{}%
```

Definition of `\@makefntext` for all other classes:

```
      \long\def\@makefntext##1{%
        \localleftbox{}%
        \let\FBeverypar@save\FBeverypar@quote
        \let\FBeverypar@quote\relax
```

```
            \ifFBFrenchFootnotes
              \@makefntextFB{##1}%
            \else
              \@makefntextORI{##1}%
            \fi
            \let\FBeverypar@quote\FBeverypar@save
            \localleftbox{\FBeveryline@quote}}%
        \fi
    }%
  }
```

\FB@legacyFootnoteSetup is executed when entering French for the first time (at \begin{document}), after possible redefinitions made by latex-lab for tagging.

```
  \def\bbl@frenchfootnotes{%
    \ifFB@mainlanguage@FR
      \ifFBnewfootnotes
      \else
        \FB@legacyFootnoteSetup
        \let\FB@legacyFootnoteSetup\relax
      \fi
    \fi}
  \addto\extrasfrench{\bbl@frenchfootnotes}
```

The next two commands are provided only with legacy code.
\StandardFootnotes may be used locally (in minipages for instance), that's why the test \ifFBFrenchFootnotes is done inside \@makefntext.

```
  \newcommand*{\FrenchFootnotes}{\FBFrenchFootnotestrue}
  \newcommand*{\StandardFootnotes}{\FBFrenchFootnotesfalse}
\fi
```

2.14 Clean up and exit

Final cleaning. The macro \ldf@finish takes care for setting the main language to be switched on at \begin{document} and resetting the category code of @ to its original value. \loadlocalcfg is redefined locally in order not to load any .cfg file for French.

```
\FBclean@on@exit
\ldf@finish\CurrentOption
\let\loadlocalcfg\FB@llc
</french>
```

3 Change History

Changes listed in reverse order (latest first) since v3.3.