

A Babel language definition file for French

frenchb.dtx v4.0d, 2025-07-25

Daniel Flipo
daniel.flipo@free.fr

Contents

1	The French language	2
1.1	Basic interface	2
1.2	Customisation	5
1.2.1	\frenchsetup	5
1.2.2	Caption names	9
1.2.3	Figure and table captions	9
1.3	Hyphenation checks	10
1.4	Changes	11
2	The code	13
2.1	Initial setup	13
2.2	Punctuation	15
2.3	Commands for French quotation marks	28
2.4	Date in French	32
2.5	Extra utilities	32
2.6	Formatting numbers	35
2.7	Caption names	38
2.8	Checks about packages' loading order	40
2.9	Setup options: key/value stuff (l3keys)	40
2.10	French lists	53
2.11	French indentation of sections	59
2.12	Formatting footnotes	59
2.13	Clean up and exit	64
3	Change History	65

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 have occurred in version 4.0a, they are listed in subsection 1.4 p. 11.

`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. 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`.

When French is loaded as the last option of Babel, `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);

¹The file described in this section has version number v4.0d and was last revised on 2025-07-25.

²*Always* use `french` as option name for the French language, former aliases `frenchb` or `francais` are *deprecated*; expect them to be removed sooner or later!

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

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 language⁴, 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: `\og some text\fg{}` is an alternative to `\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 is meant to be used inside an environment or a group.

`\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

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

⁵Or even «~guillemets~»...

«*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 1^{er} 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 3^{es}). 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 1°, 2°, 3°, 4°. `\FrenchEnumerate{6}` prints 6°.
6. Abbreviations for “Numéro(s)” and “numéro(s)” (N° N^{os} n° and n^{os}) 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 *T_EXbook* 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 ‘1^{er} 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 by 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.

`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 is 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 (the 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 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.

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:

<p>Text starting at ‘parindent’ \leq Leftmargin</p> <ul style="list-style-type: none"> — first item running on two lines or more... — first second level item on two lines... — next one... — second item... 	<p>Text starting at ‘parindent’ \leq Leftmargin</p> <ul style="list-style-type: none"> — first item running on two lines or more... — first second level item on two lines... — next one... — second item...
Default French layout	With ListItemsAsPar=true

StandardListSpacing=true (false*)⁶; `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.

StandardItemizeEnv=true (false*); `babel-french` redefines the `itemize` environment to suppress any vertical space between items of `itemize` lists in French and customises left margins. Setting this option to **true** reverts to the standard definition of `itemize`.

StandardEnumerateEnv=true (false*); `babel-french` redefines `enumerate` and `description` environments to make left margins match those of the French version of `itemize` lists. Setting this option to **true** reverts to the standard definition of `enumerate` and `description`.

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

⁶This 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`.

`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. This option is just a shorthand setting all four options `StandardListSpacing=true`, `StandardItemizeEnv=true`, `StandardEnumerateEnv=true` and `StandardItemLabels=true`.

`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}`

`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 `lwrap` (v. 0.37 and up) is fully

compatible with `babel-french` for translating PDFLaTeX or XeLaTeX files to HTML.

`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.

`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 (`~`)⁷ in French; when set to true, this option redefines `\npthousandsep` as a thin space (`\FBthinspace`).

`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.

⁷Actually without stretch nor shrink.

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 up-casing of superscripts (for instance when they are moved to page headers). Making this option **false** will disable this behaviour (not recommended).

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 T_EX 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 4.0?

`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 an 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 occurring in case `\spaceskip` is not null has been fixed.

Note on PDF tagging: this project requires a complete redesign of lists based on templates. The new lists templates, still experimental, are incompatible with `babel-french` lists' customisation, which is consequently disabled when tagging is enabled. A warning is issued in the `.log`. See <https://github.com/latex3/tagging-project/issues/694> for more information. I plan to get `babel-french` lists' customisation working again asap (hopefully with the next LaTeX release 2025/10/01).

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` have been removed: footnotes inside minipages are now numbered the standard way with both new and aold 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.40.

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 (`13keys`). 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.

```
1 <#french>
2 \LdfInit\CurrentOption{FBclean@on@exit}
```

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

```
3 \let\bbl@tempa\relax
4 \begingroup\expandafter\expandafter\expandafter\endgroup
5 \expandafter\ifx\csname luatexversion\endcsname\relax
6   \input french3.1df\relax
7   \let\bbl@tempa\endinput
8 \fi
9 \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:

```
10 \def\fb@error#1#2{%
11   \begingroup
12     \newlinechar=`^^J
13     \def\{^^J(french.1df) }%
14     \errhelp{#2}\errmessage{\{#1^^J}%
15   \endgroup}
16 \def\fb@warning#1{%
17   \begingroup
18     \newlinechar=`^^J
19     \def\{^^J(french.1df) }%
20     \message{\{#1^^J}%
21   \endgroup}
22 \def\fb@info#1{%
23   \begingroup
24     \newlinechar=`^^J
25     \def\{^^J}%
26     \wlog{#1}%
27   \endgroup}
```

Quit if the LuaTeX engine is too old.

```
28 \ifnum\luatexversion<100
29   \ifx\PackageWarning\@undefined
30     \fb@warning{Please upgrade LuaTeX to version 1.1 or above!}%
31     Aborting.}%
32   \else
33     \PackageWarning{french.1df}{Please upgrade LuaTeX
```

```

34         to version 1.1 or above!\MessageBreak
35         Aborting. Reported}%
36     \fi
37 \let\bbl@tempa\endinput
38 \fi
39 \bbl@tempa

```

Quit if Babel's version is less than 3.9i.

```

40 \let\bbl@tempa\relax
41 \ifdefined\babeltags
42 \else
43     \let\bbl@tempa\endinput
44     \ifdefined\PackageError
45         \PackageError{french.ldf}
46             {babel-french requires babel v.24.1.\MessageBreak
47             Aborting here}
48             {Please upgrade Babel!}
49     \else
50         \fb@error{babel-french requires babel v.24.1.\\
51             Aborting here}
52             {Please upgrade Babel!}
53     \fi
54 \fi
55 \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@<language>` also for eTeX, LuaTeX and XeTeX formats which set `\lang@<language>`.

```

56 \def\FB@nopatterns{%
57     \ifdefined\l@nohyphenation
58         \addialect\l@french\l@nohyphenation
59         \edef\bbl@nulllanguage{\string\language=nohyphenation}%
60     \else
61         \edef\bbl@nulllanguage{\string\language=0}%
62         \addialect\l@french0
63     \fi
64     \@nopatterns{French}}
65 \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.

```

66 \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.

```

67 \newif\ifLaTeXe
68 \let\bbl@tempa\relax
69 \ifdefined\magnification

```

```

70 \else
71   \ifdefined\@compatibilitytrue
72     \LaTeXettrue
73   \else
74     \PackageError{french.ldf}
75       {LaTeX-2.09 format is no longer supported.\MessageBreak
76       Aborting here}
77     {Please upgrade to LaTeX2e!}
78   \let\bbl@tempa\endinput
79   \fi
80 \fi
81 \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**.

```
82 \newif\ifFBfrench
```

\extrasfrench 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.

```

83 \def\extrasfrench{%
84   \FBfrenchtrue
85   \babel@savevariable{\lccode"27}%
86   \lccode"27="27
87   \babel@savevariable{\lccode"2019}%
88   \lccode"2019="2019
89   \bbl@frenchspacing
90 }
91 \def\noextrasfrench{\FBfrenchfalse \bbl@nonfrenchspacing}

```

2.2 Punctuation

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

\FBguillspace In French high punctuation characters (: ; ! ?) and guillemets require some space to be added before them (: ; ! ? » ›) or after them («, ‹). Following the I.N. specifications, **\FBcolonspace** 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.

```

92 \newcommand*{\FBguillspace}{\hskip .8\fontdimen2\font
93                               plus .3\fontdimen3\font
94                               minus .8\fontdimen4\font \relax}
95 \newcommand*{\FBcolonspace}{\space}
96 \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 arguments⁸: 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”.

```

97 \ifLaTeXe
98   \newcommand*{\FBsetspaces}[5][french]{%
99     \@namedef{FB#2space}{\hskip #3\fontdimen2\font
100                          plus #4\fontdimen3\font
101                          minus #5\fontdimen4\font \relax}}
102   \@onlypreamble\FBsetspaces
103 \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}`.

```

104 \ifLaTeXe
105   \AddToHook{env/document/before}{%
106     \set@glue@table{colon}%
107     \set@glue@table{thin}%
108     \set@glue@table{guill}%
109   }
110 \fi

```

This code is for Plain: load `ltxlua.tex` if it hasn’t been loaded before Babel.

```

111 \ifdefined\newluafunction\else
112   \input ltxlua.tex
113 \fi

```

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

`\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).

⁸The former optional `lang` argument no longer has any effect.

`\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**.

```

114 \newattribute\FB@spacing      \FB@spacing=\@ne
115 \newattribute\FB@addDPspace  \FB@addDPspace=\@ne
116 \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.

```

117 \newcommand*\set@glue@table}[1]{%
118   \directlua {
119     local s = token.get_meaning("FB#1space")
120     local t = FBget_glue(s)
121     if t then
122       FBsp.#1.gl = t
123       if FBsp.#1.gl[1] > 0.6 then
124         FBsp.#1.ch = 0xA0
125       elseif FBsp.#1.gl[1] > 0.2 then
126         FBsp.#1.ch = 0x202F
127       else
128         FBsp.#1.ch = 0x200B
129       end
130     else
131       texio.write_nl('term and log', '')
132       texio.write_nl('term and log',
133         '*** french.ldf warning: Unexpected syntax in FB#1space,')
134       texio.write_nl('term and log',
135         '*** french.ldf warning: LuaTeX table FBsp unchanged.')
136       texio.write_nl('term and log',
137         '*** french.ldf warning: Consider using FBsetspaces to ')
138       texio.write('term and log', 'customise FB#1space.')
139       texio.write_nl('term and log', '')
140     end
141   }%
142 }
143 </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).

```

144 <*\lua>
145 local FB_punct_thin =
146   {[string.byte("!")] = true,
147    [string.byte("?")] = true,
148    [string.byte(";")] = true}
149 local FB_punct_thick =
150   {[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 ‘»’.

```

151 local FB_punct_left =
152   {[string.byte("!")] = true,
153    [string.byte("?")] = true,
154    [string.byte(";")] = true,
155    [string.byte(":")] = true,
156    [0x14] = true,
157    [0xBB] = true,
158    [0x203A] = true}
159 local FB_punct_right =
160   {[0x13] = true,
161    [0xAB] = true,
162    [0x2039] = true}

```

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

```

163 local FB_punct_null =
164   {[string.byte("!")] = true,
165    [string.byte("?")] = true,
166    [string.byte("[")] = true,
167    [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.

```

168   [0xA0] = true,
169   [0x202F] = true}
170 local FB_guil_null =
171   {[0xA0] = true,
172   [0x202F] = true}

```

Local definitions for nodes:

```

173 local new_node = node.new
174 local copy_node = node.copy

```

```

175 local node_id      = node.id
176 local HLIST        = node_id("hlist")
177 local TEMP         = node_id("temp")
178 local DISC         = node_id("disc")
179 local KERN         = node_id("kern")
180 local GLUE         = node_id("glue")
181 local GLYPH        = node_id("glyph")
182 local PENALTY      = node_id("penalty")
183 local nobreak      = new_node(PENALTY)
184 nobreak.penalty    = 10000
185 local nbspace      = new_node(GLYPH)
186 local insert_node_before = node.insert_before
187 local insert_node_after  = node.insert_after
188 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.

```

189 function FBget_glue(toks)
190   local t = nil
191   local f = string.match(toks,
192     "[^%w]hskip%s*([%d%.]*)%s*[^%w]fontdimen 2")
193   if f == "" then f = 1 end
194   if tonumber(f) then
195     t = {tonumber(f), 0, 0}
196     f = string.match(toks, "plus%s*([%d%.]*)%s*[^%w]fontdimen 3")
197     if f == "" then f = 1 end
198     if tonumber(f) then
199       t[2] = tonumber(f)
200       f = string.match(toks, "minus%s*([%d%.]*)%s*[^%w]fontdimen 4")
201       if f == "" then f = 1 end
202       if tonumber(f) then
203         t[3] = tonumber(f)
204       end
205     end
206   elseif string.match(toks, "[^%w]F?B?thinspace") then
207     t = {0.5, 0, 0}
208   elseif string.match(toks, "[^%w]space") then
209     t = {1, 1, 1}
210   end
211   return t
212 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**.

```

213 FBsp = {}
214 FBsp.thin = {}
215 FBsp.thin.gl = {.5, 0, 0}
216 FBsp.thin.ch = 0x202F
217 FBsp.colon = {}
218 FBsp.colon.gl = {1, 1, 1}
219 FBsp.colon.ch = 0xA0
220 FBsp.guill = {}
221 FBsp.guill.gl = {.8, .3, .8}
222 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`. `\spaceskip`, when not null, replaces the inter-word space (in `raggedright` env. `f.i.`). This is now taken into account.

```

223 local font_table = {}
224 local function new_glue_scaled (fid,table)
225   if fid > 0 and table[1] then
226     local fp = font_table[fid]
227     if not fp then
228       local ft = font.getfont(fid)
229       if ft then
230         font_table[fid] = ft.parameters
231         fp = font_table[fid]
232       end
233     end
234     local gl = new_node(GLUE,0)

```

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

```

235   if fp then
236     local spaceskip
237     spaceskip = tex.get(tex.spaceskip)
238     local spskip = spaceskip.width
239     if spaceskip and spskip and spskip > 0 then
240       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
272       if FRspacing and fid > 0 then
273         if FB_punct_left[char] then
274           local prev = item.prev
275           local prev_id, prev_subtype, prev_char
276           if prev then
277             prev_id = prev.id
278             prev_subtype = prev.subtype
279             if prev_id == GLYPH then
280               prev_char = prev.char
281             end
282           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.

```

283         local is_glue = prev_id == GLUE
284         local glue_wd
285         if is_glue then
286             glue_wd = prev.width
287         end
288         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) precedes 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.

```

289         if FB_punct_thin[char] or FB_punct_thick[char] then
290             local SBDP = has_attribute(item, addDPspace)
291             local auto = SBDP and SBDP > 0
292             if FB_punct_thick[char] and auto then
293                 local next = item.next
294                 local next_id
295                 if next then
296                     next_id = next.id
297                 end
298                 if next_id and
299                     (next_id == GLYPH or next_id == DISC) then
300                     auto = false
301                 end
302             end
303             if auto then
304                 if (prev_char and FB_punct_null[prev_char]) or
305                     (is_glue and glue_wd  $\leq$  1) or
306                     (prev_id == HLIST and prev_subtype == 3) or
307                     (prev_id == TEMP) then
308                     auto = false
309                 end
310             end
311             local fbglue
312             local t

```

```

313         if FB_punct_thick[char] then
314             t = FBsp.colon.gl
315             nbspace.char = FBsp.colon.ch
316         else
317             t = FBsp.thin.gl
318             nbspace.char = FBsp.thin.ch
319         end
320         fbglue = new_glue_scaled(fid, t)

```

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

```

321         if (realglue or auto) and fbglue then
322             if realglue then
323                 head = remove_node(head,prev,true)
324             end
325             if (FRucsNBSP) then
326                 nbspace.font = fid
327                 nbspace.attr = attr
328                 insert_node_before(head,item,copy_node(nbspace))
329             else
330                 nobreak.attr = attr
331                 fbglue.attr = attr
332                 insert_node_before(head,item,copy_node(nobreak))
333                 insert_node_before(head,item,copy_node(fbglue))
334             end
335         end

```

Let's consider '»' and '›' now (the only remaining glyphs of `FB_punct_left` class): we just have to remove any *glue* possibly preceding 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.

```

336         else
337             local addgl = (prev_char and
338                           not FB_guil_null[prev_char])
339                           or
340                           (not prev_char and
341                             prev_id ~= TEMP and
342                             not (prev_id == HLIST and
343                                 prev_subtype == 3)
344                           )

```

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

```

345         if is_glue and glue_wd ≤ 1 then
346             addgl = false
347         end
348         local t = FBsp.guill.gl
349         nbspace.char = FBsp.guill.ch

```

```

350         local fbglue = new_glue_scaled(fid, t)
351         if addgl and fbglue then
352             if is_glue then
353                 head = remove_node(head,prev,true)
354             end
355             if (FRucsNBSP) then
356                 nbspace.font = fid
357                 nbspace.attr = attr
358                 insert_node_before(head,item,copy_node(nbspace))
359             else
360                 nobreak.attr = attr
361                 fbglue.attr = attr
362                 insert_node_before(head,item,copy_node(nobreak))
363                 insert_node_before(head,item,copy_node(fbglue))
364             end
365         end
366     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*.

```

367         elseif FB_punct_right[char] then
368             local next = item.next
369             local next_id, next_subtype, next_char, nextnext, kern_wd
370             if next then
371                 next_id = next.id
372                 next_subtype = next.subtype

```

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

```

373             if next_id == PENALTY then
374                 nextnext = next.next
375                 if nextnext and nextnext.id == GLUE then
376                     head = remove_node(head,nextnext,true)
377                     head = remove_node(head,next,true)
378                     next = item.next
379                     if next then
380                         next_id = next.id
381                         next_subtype = next.subtype
382                         if next_id == GLYPH then
383                             next_char = next.char
384                         end
385                     end
386                 end
387             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}~»):

```

388         if next_id = KERN then
389             kern_wd = next.kern
390             if kern_wd = 0 then
391                 nextnext = next.next
392                 if nextnext then
393                     next = nextnext
394                     next_id = nextnext.id
395                     next_subtype = nextnext.subtype
396                     if next_id = PENALTY then
397                         nextnext = next.next
398                         if nextnext and nextnext.id = GLUE then
399                             head = remove_node(head,next,true)
400                             head = remove_node(head,nextnext,true)
401                             next = item.next
402                             if next then
403                                 next_id = next.id
404                                 next_subtype = next.subtype
405                             end
406                         end
407                     end
408                 end
409             end
410         end
411         if next_id = GLYPH then
412             next_char = next.char
413         end
414     end
415     local is_glue = next_id = GLUE
416     if is_glue then
417         glue_wd = next.width
418     end

```

The addgl flag only depends on next_char and is_glue:

```

419         local addgl = (next_char and not FB_guil_null[next_char])
420                     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).

```

421         if is_glue and glue_wd = 0 then
422             addgl = false
423         end
424         local fid = item.font
425         local t = FBsp.guill.gl
426         nbspace.char = FBsp.guill.ch
427         local fbglue = new_glue_scaled(fid, t)

```

```

428         if addgl and fbglue then
429             if is_glue then
430                 head = remove_node(head,next,true)
431             end
432             if (FRucsNBSP) then
433                 nbspace.font = fid
434                 nbspace.attr = attr
435                 insert_node_after(head, item, copy_node(nbspace))
436             else
437                 nobreak.attr = attr
438                 fbglue.attr = attr
439                 insert_node_after(head, item, copy_node(fbglue))
440                 insert_node_after(head, item, copy_node(nobreak))
441             end
442         end
443     end
444 end
445 end
446 end
447 return head
448 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 hyphenated on the first ‘-’, never on the second one. It increases the hyphen penalty to 10000 on the second ‘-’.

```

449 local FB_t =
450     {[0x74] = true,
451     [0x54] = true}
452 local function euphonic_t (head)
453     for item in node.traverse_id(DISC, head) do
454         if item.subtype == 2 then
455             local next = item.next
456             local lang
457             local nnext
458             if next and next.id == GLYPH and FB_t[next.char] then
459                 lang = next.lang
460                 nnext = next.next
461             end
462             if lang == FR and nnext and
463                 nnext.id == DISC and nnext.subtype == 2 then
464                 nnext.penalty = 10000
465             end
466         end
467     end
468     return head
469 end
470 return french_punctuation, euphonic_t

```

471 </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`).

```

472 < *french>
473 \def\activate@luacode{%
474   \directlua{%
475     FR = \the\l@french ;
476     local path = kpse.find_file("frenchb.lua", "lua")
477     if path then
478       local f1, f2 = dofile(path)
479       luatexbase.add_to_callback("kerning",
480         f1, "frenchb.french_punctuation")
481       luatexbase.add_to_callback("pre_linebreak_filter",
482         f2, "frenchb.euphonic_t")
483     else
484       texio.write_nl('')
485       texio.write_nl('*****')
486       texio.write_nl('Error: frenchb.lua not found.')
487       texio.write_nl('*****')
488       texio.write_nl('')
489     end
490   }%
491 }
```

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

```

492 \newif\ifFBAutoSpacePunctuation \FBAutoSpacePunctuationtrue
```

`\AutoSpaceBeforeFDP` `\autospace@beforeFDP` and `\noautospace@beforeFDP` are internal commands. `\NoAutoSpaceBeforeFDP` `\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).

```

493 \def\autospace@beforeFDP{\FB@addDPspace=\@ne \relax}
494 \def\noautospace@beforeFDP{\FB@addDPspace=\@z \relax}
495 \ifLaTeXe
496   \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
497     \FBAutoSpacePunctuationtrue}
498   \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
```

```

499 \FBAutoSpacePunctuationfalse}
500 \AtEndOfPackage{\AutoSpaceBeforeFDP}
501 \else
502 \let\AutoSpaceBeforeFDP\autospace@beforeFDP
503 \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
504 \AutoSpaceBeforeFDP
505 \fi

```

\rmfamilyFB In LaTeX2e **\ttfamily** (and hence **\texttt**) will be redefined ‘AtBeginDocument’ as **\sffamilyFB** **\ttfamilyFB** so that no space is added before the four ; : ! ? characters, even if **\ttfamilyFB** **AutoSpacePunctuation** is **true**. When **AutoSpacePunctuation** is **false**, the eventually typed spaces are left unchanged (not turned into thin spaces, no penalty added). **\rmfamily** and **\sffamily** need to be redefined also (**\ttfamily** is not always used inside a group, its effect can be cancelled by **\rmfamily** or **\sffamily**).

These redefinitions can be canceled if necessary, for instance to recompile older documents, see option **OriginalTypewriter** below.

To be consistent with what is done for the ; : ! ? characters, **\ttfamilyFB** also switches off insertion of spaces inside French guillemets. This is also a workaround for the weird behaviour of these characters in verbatim mode.

```

506 \ifLaTeXe
507 \NewDocumentCommand\ttfamilyFB{}{\FB@spacing=\z@ \ttfamilyORI}
508 \NewDocumentCommand\rmfamilyFB{}{\FB@spacing=\@ne \rmfamilyORI}
509 \NewDocumentCommand\sffamilyFB{}{\FB@spacing=\@ne \sffamilyORI}
510 \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**.

```

511 \providecommand\texorpdfstring[2]{#1}
512 \DeclareRobustCommand{\NoAutoSpacing}{%
513 \texorpdfstring{\FB@spacing=\z@}{}}%
514 }

```

2.3 Commands for French quotation marks

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

```

\guillemotright \guilsinglleft 515 \ifLaTeXe
\guilsinglright 516 \else
\textquoteddblleft 517 \def\guillemotleft{\char"00AB}
\textquoteddblright 518 \def\guillemotright{\char"00BB}
519 \def\textquotedblleft{\char"201C}
520 \def\textquotedblright{\char"201D}

```

```

521 \def\guilsinglleft{{\char"2029}}
522 \def\guilsinglright{{\char"203A}}
523 \let\xspace\relax
524 \fi

```

\og 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.

```

525 \newcommand*{\og}{\textquotedblleft}
526 \newcommand*{\fg}{\textquotedblright}
527 \texorpdfstring{\ifdim\lastskip>z@\unskip\fi\textquotedblright\xspace}%
528 \textquotedblright\xspace}%
529 }

```

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).

```

530 \newcommand*{\FB@og}{\texorpdfstring{\guillemotleft}%
531 \guillemotleft\space}}
532 \newcommand*{\FB@fg}{\texorpdfstring{\ifdim\lastskip>z@\unskip\fi
533 \guillemotright}{\space\guillemotright}}
534 \def\bbl@frenchguillemets{\def\og{\FB@og}%
535 \def\fg{\FB@fg\xspace}}
536 \addto\extrasfrench{\babel@save\og \babel@save\fg
537 \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...

```

538 \newcommand*{\@ogi}{\ifmmode\hbox{\guillemotleft}\else\guillemotleft\fi}
539 \newcommand*{\@fgi}{\ifmmode\hbox{\guillemotright}\else\guillemotright\fi}
540 \newcommand*{\ogii}{\ifFBInnerGuillSingle \guilsinglleft
541 \else \textquotedblleft
542 \fi}
543 \newcommand*{\fgii}{\ifFBInnerGuillSingle \guilsinglright
544 \else \textquotedblright
545 \fi}
546 \newcommand*{\@ogii}{\ifmmode\hbox{\ogii}\else\ogii\fi}
547 \newcommand*{\@fgii}{\ifmmode\hbox{\fgii}\else\fgii\fi}

```

and the needed technical stuff to handle options:

```

548 \newcount\FBguill@level
549 \newtoks\FBbold@everypar

```

`\FB@addquote@everypar` was borrowed from `csquotes.sty`.

```

550 \def\FB@addquote@everypar{%
551   \let\FBnew@everypar\everypar
552   \FBold@everypar=\expandafter{\the\everypar}%
553   \FBnew@everypar={\the\FBbold@everypar\FBeverypar@quote}%
554   \let\everypar\FBbold@everypar
555   \let\FB@addquote@everypar\relax
556 }
557 \newif\ifFBcloseguill \FBcloseguilltrue
558 \newif\ifFBinnerguillsingle
559 \def\FBguillopen{\guillemotleft}
560 \def\FBguillclose{\guillemotright}
561 \let\FBguillnone\empty
562 \let\FBeveryparguill\FBguillopen
563 \let\FBverylineguill\FBguillnone
564 \let\FBeverypar@quote\relax
565 \let\FBveryline@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.

```

566 \ifLaTeXe
567   \DeclareRobustCommand\frquote{%
568     \texorpdfstring{\@ifstar{\FBcloseguillfalse\fr@quote}%
569                       {\FBcloseguilltrue \fr@quote}}%
570     {\bm@fr@quote}%
571   }
572   \newcommand{\bm@fr@quote}[1]{« #1 »}
573 \else
574   \newcommand\frquote[1]{\fr@quote{#1}}
575 \fi

```

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

```

576 \newcommand{\fr@quote}[1]{%
577   \leavevmode
578   \advance\FBguill@level by \@ne
579   \ifcase\FBguill@level
580     \or

```

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

```

581   \ifx\FBeveryparguill\FBguillnone
582   \else
583     \def\FBeverypar@quote{\FBeveryparguill}%

```

```

584     \FB@addquote@everypar
585     \fi
586     \@ogi #1\@fgi
587 \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.

```

588     \ifx\FBverylineguill\FBguillopen
589     \def\FBveryline@quote{\guillemotleft\FBguillspace}%
590     \localleftbox{\FBveryline@quote}%
591     \let\FBverypar@quote\relax
592     \@ogi #1\ifFBcloseguill\@fgi\fi
593 \else
594     \ifx\FBverylineguill\FBguillclose
595     \def\FBveryline@quote{\guillemotright\FBguillspace}%
596     \localleftbox{\FBveryline@quote}%
597     \let\FBverypar@quote\relax
598     \@ogi #1\ifFBcloseguill\@fgi\fi
599 \else

```

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

```

600     \let\FBverypar@quote\relax
601     \ifFBInnerGuillSingle
602     \ifx\FBveryparguill\FBguillopen
603     \def\FBverypar@quote{\guilsinglleft\FBguillspace}%
604     \fi
605     \ifx\FBveryparguill\FBguillclose
606     \def\FBverypar@quote{\guilsinglright\FBguillspace}%
607     \fi
608     \fi
609     \@ogii #1\@fgii
610 \fi
611 \fi
612 \else

```

Warn if `\FBguill@level > 2`:

```

613     \ifx\PackageWarning\@undefined
614     \fb@warning{\noexpand\frquote\space handles up to
615                two levels.\\ Quotation not printed.}%
616 \else
617     \PackageWarning{french.ldf}{%
618         \protect\frquote\space handles up to two levels.
619         \MessageBreak Quotation not printed. Reported}
620 \fi
621 \fi

```

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

```

622 \global\advance\FBguill@level by \m@ne
623 \ifcase\FBguill@level \global\let\FBeverypar@quote\relax
624 \or \gdef\FBeverypar@quote{\FBeveryparguill}%
625 \global\let\FBeverylin@quote\empty
626 \ifx\FBeverylin@guill\FBguillnone\else\localleftbox{}\fi
627 \fi
628 }

```

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.

```

629 \newcommand*{\NoEveryParQuote}{\let\FBeveryparguill\FBguillnone}

```

2.4 Date in French

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

```

630 \StartBabelCommands*{\french}{\date}
631 [unicode, fontenc=TU EU1 EU2, charset=utf8]
632 \SetStringLoop{month#1name}{%
633     janvier,février,mars,avril,mai,juin,juillet,%
634     août,septembre,octobre,novembre,décembre}
635 \SetString\today{\FB@date{\year}{\month}{\day}}
636 \EndBabelCommands

```

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

```

637 \newcommand*{\FB@date}[3]{%
638     {\number#3}\ifnum1=#3{\ier}\fi\FBdatespace
639     \csname month\romannumeral#2name\endcsname
640     \ifx#1\@empty\else\FBdatespace\number#1\fi}}
641 \newcommand*{\FBdatebox}{\hbox}
642 \newcommand*{\FBdatespace}{\space}
643 \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 '1^{er}'.

\fup When a font has built-in superscripts, the best thing to do is to just use them, otherwise `\fup` provides an alternative which typesets superscripts slightly smaller and higher. Scaling is done using package `scalegnt` 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.

```
644 \newif\ifFBFrenchSuperscripts      \FBFrenchSuperscriptstrue
645 \newif\ifFBLowercaseSuperscripts   \FBLowercaseSuperscriptstrue
646 \newdimen\FB@Mht
647 \ifLaTeXe
648   \AtEndOfPackage{\RequirePackage{scalefont}}
```

`\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{}`.

```
649 \newcommand*{\FBsupR}{-0.12}
650 \newcommand*{\FBsupS}{0.65}
651 \newcommand*{\FB@lc}[1]{\MakeLowercase{#1}}
652 \NewDocumentCommand\fup{ m }{%
653   \settoheight{\FB@Mht}{M}%
654   \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
655   \addtolength{\FB@Mht}{-\FBsupS ex}%
656   \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
657 }
```

Poor man’s definition of `\up` for Plain.

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

\ieme Some handy macros for those who don’t know how to abbreviate ordinals:

```
\ier 661 \def\ieme{\up{e}\xspace}
\iere 662 \def\iemes{\up{es}\xspace}
\iemes 663 \def\ier{\up{er}\xspace}
\iers 664 \def\iers{\up{ers}\xspace}
\ieres 665 \def\iere{\up{re}\xspace}
666 \def\ieres{\up{res}\xspace}
```

\FBmedkern Configurable kerns `\FBmedkern`, and `\FBthickkern` suitable for HTML translation.
\FBthickkern 667 \newcommand*{\FBmedkern}{\kern+.2em}
668 \newcommand*{\FBthickkern}{\kern+.3em}

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

```
\fprimo 669 \newcommand*{\FrenchEnumerate}[1]{%
\nos 670   #1\textorpdfstring{\up{o}\FBthickkern}{\textdegree\space}}
\nos
\no
\no
```

```

671 \newcommand*{\FrenchPopularEnumerate}[1]{%
672     #1\texorpdfstring{\up{o}}\FBthickkern}\textdegree\space}}

```

Typing `\primo` should result in ‘°’ (except in bookmarks where `\textdegree` is used instead of o-superior),

```

673 \def\primo{\FrenchEnumerate1}
674 \def\secundo{\FrenchEnumerate2}
675 \def\tertio{\FrenchEnumerate3}
676 \def\quarto{\FrenchEnumerate4}

```

while typing `\fprimo` gives ‘º’ (except in bookmarks where `\textdegree` is used instead),.

```

677 \def\fprimo{\FrenchPopularEnumerate1}
678 \def\fsecundo{\FrenchPopularEnumerate2}
679 \def\ftertio{\FrenchPopularEnumerate3}
680 \def\fquarto{\FrenchPopularEnumerate4}

```

Let’s provide four macros for the common abbreviations of “Numéro”. In bookmarks º is used instead of o-superior.

```

681 \DeclareRobustCommand*{\No}{%
682     \texorpdfstring{N\up{o}}\FBmedkern}\N\textdegree\space}}
683 \DeclareRobustCommand*{\no}{%
684     \texorpdfstring{n\up{o}}\FBmedkern}\n\textdegree\space}}
685 \DeclareRobustCommand*{\Nos}{%
686     \texorpdfstring{N\up{os}}\FBmedkern}\N\textdegree\space}}
687 \DeclareRobustCommand*{\nos}{%
688     \texorpdfstring{n\up{os}}\FBmedkern}\n\textdegree\space}}

```

\bname These commands are meant to easily enter family names (in small capitals for the latter) while avoiding 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.

```

689 \ifLaTeXe
690     \DeclareRobustCommand*{\bname}[1]{%
691         \texorpdfstring{\leavevmode\begingroup\kern0pt #1\endgroup}\{#1}%
692     }
693     \DeclareRobustCommand*{\bsc}[1]{%
694         \texorpdfstring{\leavevmode\begingroup\kern0pt \scshape #1\endgroup}%
695             {\textsc{#1}}}%
696     }
697 \else
698     \newcommand*{\bname}[1]{\leavevmode\begingroup\kern0pt #1\endgroup}
699     \let\bsc\bname
700 \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`.
`\degre` can be accessed by the command `\r{}` for ring accent.

```

701 \providecommand*\textbackslash{{\char"005C}}
702 \providecommand*\textasciicircum{{\char"005E}}
703 \providecommand*\textasciitilde{{\char"007E}}
704 \providecommand*\degre{{}^}
705 \providecommand*\degres{{}^\circ}
706 \providecommand*\boi{{\textbackslash}}
707 \providecommand*\circonflexe{{\textasciicircum}}
708 \providecommand*\tild{{\textasciitilde}}
709 \newcommand*\at{{@}}

```

2.6 Formatting numbers

`\StandardMathComma` As mentioned in the T_EXbook p. 134, the comma is of type `\mathpunct` in math mode:
`\DecimalMathComma` 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.

```

710 \newif\ifFB@icomma
711 \newcount\mc@charclass
712 \newcount\mc@charfam
713 \newcount\mc@charslot
714 \newcount\std@mcc
715 \newcount\dec@mcc
716 \mc@charclass=\Umathcharclass`,
717 \newcommand*\dec@math@comma{%
718   \mc@charfam=\Umathcharfam`,
719   \mc@charslot=\Umathcharslot`,
720   \Umathcode`= \mc@charfam \mc@charslot
721 }
722 \newcommand*\std@math@comma{%
723   \mc@charfam=\Umathcharfam`,
724   \mc@charslot=\Umathcharslot`,
725   \Umathcode`= \mc@charclass \mc@charfam \mc@charslot
726 }
727 \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`.

```

728 \newif\if@FBpreamble
729 \ifLaTeXe \@FBpreambletrue \fi

```

```

730 \newif\if@preamble@DecimalMathComma
731 \newcommand*{\DecimalMathComma}{%
732   \if@FBpreamble \@preamble@DecimalMathComma true
733   \else
734     \ifFB@icomma
735       \PackageWarning{french.lfd}{%
736         icomma package loaded, \protect\DecimalMathComma\MessageBreak
737         does nothing. Reported}%
738     \else
739       \ifFBfrench
740         \dec@math@comma
741         \let\dec@m@c\dec@math@comma
742         \expandafter\addto\csname extras\language\endcsname
743         {\dec@m@c}%
744       \fi
745     \fi
746   \fi
747 }
748 \newcommand*{\StandardMathComma}{%
749   \ifFB@icomma
750     \PackageWarning{french.lfd}{%
751       icomma package loaded, \protect\StandardMathComma\MessageBreak
752       does nothing. Reported}%
753   \else
754     \ifFBfrench
755       \std@math@comma
756       \let\dec@m@c\relax
757     \fi
758   \fi
759 }

```

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

```

760 \ifLaTeXe\else
761   \addto\noextrasfrench{\std@math@comma}
762 \fi

```

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

```

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

```

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

```

765 \activate@luacode
766 \let\FBstop@here\relax
767 \def\FBclean@on@exit{%

```

```

768 \let\ifLaTeXe\iffalse
769 \let\LaTeXetrue\undefined
770 \let\LaTeXefalse\undefined
771 \let\FB@llc\loadlocalcfg
772 \let\loadlocalcfg\@gobble}
773 \ifx\magnification\@undefined
774 \else
775 \def\FBstop@here{%
776 \FBclean@on@exit
777 \ldf@finish\CurrentOption
778 \let\loadlocalcfg\FB@llc
779 \endinput}
780 \fi
781 \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 a French. Can be canceled anytime by `\StandardMathComma`.

```

782 \AddToHook{env/document/before}{%
783 \@FBpreamblefalse
784 \IfPackageLoadedTF{icomma}%
785 {\FB@icommatrue
786 \if@preamble@DecimalMathComma
787 \PackageWarning{french.ldf}{%
788 icomma package loaded, \protect\DecimalMathComma%
789 \MessageBreak does nothing. Reported}%
790 \fi
791 }%
792 {\if@preamble@DecimalMathComma
793 \ifFB@mainlanguage@FR \dec@math@comma \fi
794 \let\dec@m@c\dec@math@comma
795 \addto\extrasfrench{\dec@m@c}%
796 \fi

```

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

```

797 \addto\noextrasfrench{\std@math@comma}%
798 }%
799 }

```

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.

```

800 \renewcommand*{\nombre}[1]{\Warning@nombre{#1}}

```

```

801 \newcommand*{\Warning@nombre}[1]{%
802   \ifdefined\numprint
803     \numprint{#1}%
804   \else
805     \PackageWarning{french.ldf}{%
806       \protect\nombre\space now relies on package numprint.sty,%
807       \MessageBreak add \protect
808       \usepackage[autolanguage]{numprint},\MessageBreak
809       see file numprint.pdf for more options.\MessageBreak
810       \protect\nombre\space called}%
811     \global\let\Warning@nombre\relax
812     {#1}%
813   \fi
814 }

815 \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).

```

816 \StartBabelCommands*{french}{captions}
817   [unicode, fontenc=TU EU1 EU2, charset=utf8]
818   \SetString{\refname}{Références}
819   \SetString{\abstractname}{Résumé}
820   \SetString{\bibname}{Bibliographie}
821   \SetString{\chaptername}{Chapitre}
822   \SetString{\prefacename}{Préface}
823   \SetString{\appendixname}{Annexe}
824   \SetString{\contentsname}{Table des matières}
825   \SetString{\listfigurename}{Table des figures}
826   \SetString{\listtablename}{Liste des tableaux}
827   \SetString{\indexname}{Index}
828   \SetString{\glossaryname}{Glossaire}
829   \SetString{\figurename}{Figure}
830   \SetString{\tablename}{Table}
831   \SetString{\pagename}{page}
832   \SetString{\seename}{voir}
833   \SetString{\alsoname}{voir aussi}
834   \SetString{\enclname}{P.~J. }
835   \SetString{\ccname}{Copie à }
836   \SetString{\headtoname}{}
837   \SetString{\proofname}{Démonstration}
838   \SetString{\partnameord}{partie}
839   \SetString{\partfirst}{Première}
840   \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`.

```

841 \SetStringLoop{ordinal#1}{%
842   \frenchpartfirst,\frenchpartsecond,Troisième,Quatrième,%
843   Cinquième,Sixième,Septième,Huitième,Neuvième,Dixième,Onzième,%
844   Douzième,Treizième,Quatorzième,Quinzième,Seizième,%
845   Dix-septième,Dix-huitième,Dix-neuvième,Vingtième}
846 \AfterBabelCommands{%
847   \NewDocumentCommand\FB@emptypart{}{\def\thepart{\unskip}}%
848   \NewDocumentCommand\FB@partname{}{%
849     \ifFBPartNameFull
850       \csname ordinal\romannumeral\value{part}\endcsname\space
851       \partnameord\FB@emptypart
852     \else
853       Partie%
854     \fi}%
855   }
856   \SetString{\partname}{\FB@partname}
857 \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`.

```

858 \providecommand*{\FBfigtabshape}{\scshape}

```

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.

```

859 \ExplSyntaxOn
860 \NewExpandableDocumentCommand\FBlower{m}
861 {
862   \str_lowercase:n {#1}
863 }
864 \ExplSyntaxOff
865 \newcommand*\FBtocpartsep{\protect\space}
866 \NewDocumentCommand\FBtocpartname{m}{%
867   \ifFBTocPartNameFull
868     \csname ordinal\FBlower{#1}\endcsname\space
869     \partnameord \FBtocpartsep \FB@emptypart
870   \else
871     Partie%
872   \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`.

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

Package `beamerarticle` should be loaded before `babel-french` to avoid list's conflicts, see p. 41.

```
874 \newif\if@FBwarning@beamerarticle
875 \IfPackageLoadedTF{beamerarticle}{}{\@FBwarning@beamerarticletrue}
876 \AddToHook{env/document/before}{%
877   \if@FBwarning@beamerarticle
878     \IfPackageLoadedTF{beamerarticle}{}%
879                                   {\@FBwarning@beamerarticlefalse}%
880   \fi
881   \if@FBwarning@beamerarticle
882     \FBWarning{Please load the "beamerarticle" package\MessageBreak
883               BEFORE babel/french; reported}%
884   \fi
885 }
```

2.9 Setup options: key/value stuff (l3keys)

Check LaTeX2e version (support for l3keys required).

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

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

```
887 \newif\ifFBnewlists
888 \newif\ifFBnewfootnotes
889 \IfPackageLoadedTF{latex-lab-testphase-block}{\FBnewliststrue}{}
890 \IfPackageLoadedTF{latex-lab-testphase-block}{\FBnewfootnotesttrue}{}%
```

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`.

```
891 \newif\ifFBShowOptions
```



```

892 \newif\ifFBStandardLayout          \FBStandardLayouttrue
893 \newif\ifFBStandardListSpacing     \FBStandardListSpacingtrue
894 \newif\ifFBListItemsAsPar
895 \newif\ifFBCompactItemize
896 \newif\ifFBStandardItemizeEnv      \FBStandardItemizeEnvtrue
897 \newif\ifFBStandardEnumerateEnv    \FBStandardEnumerateEnvtrue
898 \newif\ifFBStandardItemLabels      \FBStandardItemLabelstrue
899 \newif\ifFBStandardLists           \FBStandardListstrue
900 \newif\ifFBIndentFirst
901 \newif\ifFBFrenchFootnotes
902 \newif\ifFBAutoSpaceFootnotes
903 \newif\ifFBOriginalTypewriter
904 \newif\ifFBThinColonSpace
905 \newif\ifFBThinSpaceInFrenchNumbers
906 \newif\ifFBUnicodeNoBreakSpaces
907 \newif\ifFBINGuillSpace
908 \newif\ifFBPartNameFull
909 \newif\ifFBTocPartNameFull
910 \newif\ifFBSmallCapsFigTabCaptions
911 \newif\ifBBCustomiseFigTabCaptions
912 \newif\ifBBSuppressWarning

```

Some specific code for the koma-script classes.

```

913 \newif\ifFB@koma
914 \ifLaTeXe
915   \@ifclassloaded{scrartcl}{\FB@komatrue}{}
916   \@ifclassloaded{scrbook}{\FB@komatrue}{}
917   \@ifclassloaded{scrreprt}{\FB@komatrue}{}
918 \fi
919 \ifFB@koma
920   \ifdefined\partformat
921     \def\FB@partformat@fix{%
922       \ifFBPartNameFull
923         \babel@save\partformat
924         \renewcommand*{\partformat}{\partname}%
925       \fi
926     \addto\extrasfrench{\FB@partformat@fix}%
927   \fi
928 \fi

```

Some of the flags must be toggled 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.

```

929 \def\FB@french{french}
930 \newif\ifFB@mainlanguage@FR
931 \AtEndOfPackage{%
932   \ifx\bb1@main@language\FB@french \FB@mainlanguage@FRtrue \fi
933   \ifFB@mainlanguage@FR
934     \@ifclassloaded{beamer}%
935     {\PackageInfo{french.1df}{%
936       No list customisation for the beamer class,%
937       \MessageBreak reported}}%
938     {\IfPackageLoadedTF{beamerarticle}%
939      {\FBStandardItemLabelsfalse
940       \FBStandardListSpacingfalse
941       \PackageInfo{french.1df}{%
942         Minimal list customisation for the beamerarticle%
943         \MessageBreak package; reported}}}%

```

Otherwise customise lists “à la française”:

```

944     {\FBStandardListSpacingfalse
945      \FBStandardItemizeEnvfalse
946      \FBStandardEnumerateEnvfalse
947      \FBStandardItemLabelsfalse}%
948   }
949   \FBIndentFirsttrue
950   \FBFrenchFootnotesttrue
951   \FBAutoSpaceFootnotesttrue
952   \FBPartNameFulltrue
953   \FBTocPartNameFulltrue
954   \FBStandardLayouttrue
955   \FBSmallCapsFigTabCaptionstrue
956 \fi
957 }

```

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

```

958 \DeclareKeys[FBsetup]
959 {
960   ShowOptions.if           = FBShowOptions
961   StandardLayout.default:n = {true}
962   StandardLayout.code       = \FBStandardLayout@setup{#1}
963   StandardListSpacing.if    = FBStandardListSpacing
964   ReduceListSpacing.ifnot   = FBStandardListSpacing
965   CompactItemize.default:n  = {true}
966   CompactItemize.code       = \FBCompactItemize@setup{#1}
967   StandardItemizeEnv.if     = FBStandardItemizeEnv
968   StandardEnumerateEnv.if   = FBStandardEnumerateEnv
969   StandardItemLabels.if     = FBStandardItemLabels

```

```

970 ItemLabels.store = \FrenchLabelItem ,
971 ItemLabeli.store = \Frlabelitemi ,
972 ItemLabelii.store = \Frlabelitemii ,
973 ItemLabeliii.store = \Frlabelitemiii ,
974 ItemLabeliv.store = \Frlabelitemiv ,
975 StandardLists.default:n = {true} ,
976 StandardLists.code = \FBStandardLists@setup{#1} ,
977 ListItemsAsPar.if = FBListItemsAsPar ,
978 IndentFirst.if = FBIndentFirst ,
979 FrenchFootnotes.if = FBFrenchFootnotes ,
980 AutoSpaceFootnotes.if = FBAutoSpaceFootnotes ,
981 AutoSpacePunctuation.if = FBAutoSpacePunctuation ,
982 OriginalTypewriter.if = FBOriginalTypewriter ,
983 ThinColonSpace.default:n = {true} ,
984 ThinColonSpace.code = \FBThinColonSpace@setup{#1} ,
985 ThinSpaceInFrenchNumbers.if = FBThinSpaceInFrenchNumbers ,
986 UnicodeNoBreakSpaces.if = FBUnicodeNoBreakSpaces ,
987 FrenchSuperscripts.if = FBFrenchSuperscripts ,
988 LowercaseSuperscripts.if = FBLowercaseSuperscripts ,
989 PartNameFull.if = FBPartNameFull ,
990 TocPartNameFull.if = FBTOCPartNameFull ,
991 CustomiseFigTabCaptions.default:n = {true} ,
992 CustomiseFigTabCaptions.code = \FBCustomiseFigTabCaptions@setup{#1} ,
993 SmallCapsFigTabCaptions.default:n = {true} ,
994 SmallCapsFigTabCaptions.code = \FBSmallCapsFigTabCaptions@setup{#1} ,
995 SuppressWarning.default:n = {true} ,
996 SuppressWarning.code = \FBSuppressWarning@setup{#1} ,
997 INGuillSpace.default:n = {true} ,
998 INGuillSpace.code = \FBINGuillSpace@setup{#1} ,
999 InnerGuillSingle.if = FBInnerGuillSingle ,
1000 EveryParGuill.default:n = {open} ,
1001 EveryParGuill.code = \FBEveryParGuill@setup{#1} ,
1002 EveryLineGuill.default:n = {open} ,
1003 EveryLineGuill.code = \FBEveryLineGuill@setup{#1} ,
1004 og.code = \FBog@setup{#1} ,
1005 fg.code = \FBfg@setup{#1} ,
1006 }

```

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 later (just before `\begin{document}`) by `\FBprocess@options`. `\frenchsetup` can only be called in the preamble.

```

1007 \newcommand*{\frenchsetup}[1]{%
1008   \SetKeys[FBsetup]{#1}%
1009 }%
1010 \@onlypreamble\frenchsetup

```

Keep the former name `\frenchbsetup` working for compatibility.

```

1011 \let\frenchbsetup\frenchsetup
1012 \@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 *explicitly* set in `\frenchsetup{}`.

```

1013 \newcommand*{\FBStandardLayout@setup}[1]%
1014   {\ifFB@mainlanguage@FR
1015     \csname FBStandardLayout#1\endcsname
1016   \else
1017     \PackageWarning{french.1df}%
1018       {Option `StandardLayout' skipped:\MessageBreak
1019       French is not babel's last option.\MessageBreak
1020       Reported}%
1021   \fi
1022   \ifFBStandardLayout
1023     \FBStandardListSpacingtrue
1024     \FBStandardItemizeEnvtrue
1025     \FBStandardItemLabelstrue
1026     \FBStandardEnumerateEnvtrue
1027     \FBIndentFirstfalse
1028     \FBFrenchFootnotesfalse
1029     \FBAutoSpaceFootnotesfalse
1030   \else
1031     \FBStandardListSpacingfalse
1032     \FBStandardItemizeEnvfalse
1033     \FBStandardItemLabelfalse
1034     \FBStandardEnumerateEnvfalse
1035     \FBIndentFirsttrue
1036     \FBFrenchFootnotestrue
1037     \FBAutoSpaceFootnotestrue
1038   \fi
1039   }
1040 \newcommand*{\FBCompactItemize@setup}[1]%
1041   {\csname FBCompactItemize#1\endcsname
1042   \ifFBCompactItemize
1043     \FBStandardItemizeEnvfalse
1044     \FBStandardEnumerateEnvfalse
1045   \else
1046     \FBStandardItemizeEnvtrue
1047     \FBStandardEnumerateEnvtrue
1048   \fi
1049   }
1050 \newcommand*{\FBStandardLists@setup}[1]%
1051   {\csname FBStandardLists#1\endcsname
1052   \ifFBStandardLists
1053     \FBStandardListSpacingtrue

```

```

1054 \FBStandardItemizeEnvtrue
1055 \FBStandardEnumerateEnvtrue
1056 \FBStandardItemLabelstrue
1057 \else
1058 \FBStandardListSpacingfalse
1059 \FBStandardItemizeEnvfalse
1060 \FBStandardEnumerateEnvfalse
1061 \FBStandardItemLabelsfalse
1062 \fi
1063 }
1064 \newcommand*{\FBThinColonSpace@setup}[1]%
1065 {\csname FBThinColonSpace#1\endcsname
1066 \ifFBThinColonSpace
1067 \renewcommand*{\FBcolonspace}{\FBthinspace}%
1068 \fi
1069 }
1070 \newcommand*{\FBSmallCapsFigTabCaptions@setup}[1]%
1071 {\csname FBSmallCapsFigTabCaptions#1\endcsname
1072 \ifFBSmallCapsFigTabCaptions
1073 \else
1074 \let\FBfigtabshape\relax
1075 \fi
1076 }
1077 \newcommand*{\FBCustomiseFigTabCaptions@setup}[1]%
1078 {\csname FBCustomiseFigTabCaptions#1\endcsname
1079 \FBWarning{Option CustomiseFigTabCaptions is *OBSOLETE*\MessageBreak
1080 The "caption" package is your friend,\MessageBreak
1081 see "frenchb.pdf" for more hints.\MessageBreak
1082 Trying to use endash though... reported
1083 }%
1084 \ifFBCustomiseFigTabCaptions
1085 \IfClassLoadedTF{memoir}%
1086 {\captiondelim{\space\textendash\space}}%
1087 {\ifFB@koma
1088 \renewcommand{\captionformat}{\space\textendash\space}%
1089 \else
1090 \IfClassLoadedTF{beamer}%
1091 {\setbeamertemplate{caption label separator}[endash]}%
1092 {\RequirePackage[labelsep=endash]{caption}}%
1093 \fi
1094 }%
1095 \fi
1096 }
1097 \newcommand*{\FBSuppressWarning@setup}[1]%
1098 {\csname FBSuppressWarning#1\endcsname
1099 \ifFBSuppressWarning

```

```

1100 \renewcommand{\FBWarning}[1]{}%
1101 \fi
1102 }
1103 \newcommand*\FBINGuillSpace@setup}[1]%
1104 {\csname FBINGuillSpace#1\endcsname
1105 \ifFBINGuillSpace
1106 \FBsetspaces{guill}{1}{1}{1}%
1107 \fi
1108 }
1109 \newcommand*\FBEveryParGuill@setup}[1]%
1110 {\expandafter\let\expandafter
1111 \FBeveryparguill\csname FBguill#1\endcsname
1112 \ifx\FBEveryparguill\FBguillopen
1113 \else\ifx\FBEveryparguill\FBguillclose
1114 \else\ifx\FBEveryparguill\FBguillnone
1115 \else
1116 \let\FBEveryparguill\FBguillopen
1117 \FBWarning{Wrong value for `EveryParGuill':
1118 try `open',\MessageBreak
1119 `close' or `none'. Reported}%
1120 \fi
1121 \fi
1122 \fi
1123 }
1124 \newcommand*\FBEveryLineGuill@setup}[1]%
1125 {\expandafter\let\expandafter
1126 \FBeverylineguill\csname FBguill#1\endcsname
1127 \ifx\FBEverylineguill\FBguillopen
1128 \else\ifx\FBEverylineguill\FBguillclose
1129 \else\ifx\FBEverylineguill\FBguillnone
1130 \else
1131 \let\FBEverylineguill\FBguillnone
1132 \FBWarning{Wrong value for `EveryLineGuill':
1133 try `open',\MessageBreak
1134 `close' or `none'. Reported}%
1135 \fi
1136 \fi
1137 \fi
1138 }

```

This option has been kept for backward compatibility but is no longer necessary as the `\FB@addGUILLspace` attribute for LuaTeX is set to one (true) by default. A warning is issued.

```

1139 \newcommand*\FBog@setup}[1]{%
1140 \FBWarning{Options og=«, fg=» are not needed with LuaTeX.%
1141 \MessageBreak Automatic spacing of « » and < > is active.%
1142 \MessageBreak Use \protect\NoAutoSpacing\space (inside a group) to%

```

```

1143 \MessageBreak cancel spacing locally. Reported }
1144 }
1145 \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.

```

1146 \newcommand*{\FBprocess@options}{%

```

Update flags if a package customising lists has been loaded, currently: enumitem, paralist, enumerate.

```

1147 \IfPackageLoadedTF{enumitem}{%
1148   \ifFBStandardItemizeEnv
1149   \else
1150     \FBStandardItemizeEnvtrue
1151     \PackageInfo{french.1df}%
1152     {Setting StandardItemizeEnv=true for\MessageBreak
1153      compatibility with enumitem package,\MessageBreak
1154      reported}%
1155   \fi
1156   \ifFBStandardEnumerateEnv
1157   \else
1158     \FBStandardEnumerateEnvtrue
1159     \PackageInfo{french.1df}%
1160     {Setting StandardEnumerateEnv=true for\MessageBreak
1161      compatibility with enumitem package,\MessageBreak
1162      reported}%
1163   \fi}}%
1164 \IfPackageLoadedTF{paralist}{%
1165   \ifFBStandardItemizeEnv
1166   \else
1167     \FBStandardItemizeEnvtrue
1168     \PackageInfo{french.1df}%
1169     {Setting StandardItemizeEnv=true for\MessageBreak
1170      compatibility with paralist package,\MessageBreak
1171      reported}%
1172   \fi
1173   \ifFBStandardEnumerateEnv
1174   \else
1175     \FBStandardEnumerateEnvtrue
1176     \PackageInfo{french.1df}%
1177     {Setting StandardEnumerateEnv=true for\MessageBreak
1178      compatibility with paralist package,\MessageBreak
1179      reported}%

```

```

1180   \fi}}}%
1181 \IfPackageLoadedTF{enumerate}{%
1182   \ifFBStandardEnumerateEnv
1183   \else
1184     \FBStandardEnumerateEnvtrue
1185     \PackageInfo{french.ldf}%
1186       {Setting StandardEnumerateEnv=true for\MessageBreak
1187         compatibility with enumerate package,\MessageBreak
1188         reported}%
1189   \fi}}}%
1190 \ifFB@mainlanguage@FR
1191 \else
1192   \ifFBStandardItemizeEnv
1193   \else
1194     \PackageWarning{french.ldf}%
1195       {babel-french will not customise lists' layout\MessageBreak
1196         when French is not the main language,\MessageBreak
1197         reported}%
1198   \fi
1199 \fi

```

When tagging is enabled, lists's customisation is currently disabled, a warning is issued.

Legacy lists are customised in `\extrafrench{}` which be called by babel later on (`\AtBeginDocument{}`).

```

1200 \ifFBnewlists
1201   \FBWarning{You requested LaTeX tagging support.\MessageBreak
1202     Babel-french's list customization is currently\MessageBreak
1203     incompatible with the new lists' implementation\MessageBreak
1204     (still experimental) required to support tagging.%
1205     \MessageBreak Babel-french's list customization is *DISABLED*%
1206     \MessageBreak when tagging is enabled (see frenchb.pdf).%
1207     \MessageBreak Reported
1208   }%
1209   \setlistindentFB
1210 \fi

```

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

```

1211 \ifFBnewfootnotes
1212   \ifdim\parindentFFN<\maxdimen
1213   \else
1214     \parindentFFN=\parindent
1215     \ifdim\parindentFFN<1.8em \parindentFFN=1.8em \fi
1216   \fi
1217   \settowidth{\FBfnindent}{\dotFFN\kernFFN}%
1218   \addtolength{\FBfnindent}{\parindentFFN}%

```



```

1219 \ifFBFrenchFootnotes
1220   \NewSocketPlug{fntext/mark}{french}
1221   {\hb@xt@ \footnotemargin{\hss\newfootnotemarkFB}}
1222   \AssignSocketPlug{fntext/mark}{french}
1223   \AddToHook{cmd/maketitle/before}
1224     {\AssignSocketPlug{fntext/mark}{default}}
1225   \AddToHook{cmd/maketitle/after}
1226     {\AssignSocketPlug{fntext/mark}{french}}
1227   \AddToHook{env/minipage/begin}
1228     {\AssignSocketPlug{fntext/mark}{default}}
1229   \AddToHook{fntext/para}{\parindent=\parindentFFN}
1230   \AddToHook{fntext/para}{\lcalleftbox{}}
1231   \AddToHook{fntext/para}{\let\FBeverypar@quote\relax}
1232 \fi
1233 \ifFBAutoSpaceFootnotes
1234   \AddToHook{fnmark/before}{\FBfnmarkspace}
1235 \fi
1236 \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 commands, we customise `\fnum@figure` and `\fnum@table` when available (not in `beamer.cls` f.i.).

```

1237 \ifx\FBfigtabshape\relax
1238 \else
1239   \ifdefined\fnum@figure
1240     \let\fnum@figureORI\fnum@figure
1241     \renewcommand{\fnum@figure}{\ifFBfrench\FBfigtabshape\fi
1242                                   \fnum@figureORI}}%
1243   \fi
1244   \ifdefined\fnum@table
1245     \let\fnum@tableORI\fnum@table
1246     \renewcommand{\fnum@table}{\ifFBfrench\FBfigtabshape\fi
1247                                   \fnum@tableORI}}%
1248   \fi
1249 \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.

```

1250 \ifFBAutoSpacePunctuation
1251   \autospace@beforeFDP
1252 \else
1253   \noautospace@beforeFDP
1254   \FBWarning{AutoSpacePunctuation should *not* be set to false%
1255   \MessageBreak in LuaTeX, unless you know what you are doing.%

```

```

1256 \MessageBreak Reported }
1257 \fi

```

When **OriginalTypewriter** is set to **false** (the default), `\ttfamily`, `\rmfamily` and `\sffamily` are redefined as `\ttfamilyFB`, `\rmfamilyFB` and `\sffamilyFB` respectively to prevent addition of automatic spaces before the four active characters in computer code.

```

1258 \ifFBOriginalTypewriter
1259 \else
1260 \NewCommandCopy\ttfamilyORI\ttfamily
1261 \NewCommandCopy\rmfamilyORI\rmfamily
1262 \NewCommandCopy\sffamilyORI\sffamily
1263 \RenewCommandCopy\ttfamily\ttfamilyFB
1264 \RenewCommandCopy\rmfamily\rmfamilyFB
1265 \RenewCommandCopy\sffamily\sffamilyFB
1266 \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.

```

1267 \IfPackageLoadedTF{numprint}%
1268 {\ifnprt@autolanguage
1269 \providecommand*\npstylefrench{}{}%
1270 \ifFBThinSpaceInFrenchNumbers
1271 \renewcommand*\FBthousandsep{\FBthinspace}%
1272 \fi
1273 \g@addto@macro\npstylefrench{\npthousandsep\FBthousandsep}%
1274 \fi
1275 }{}%

```

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”.

```

1276 \ifFBFrenchSuperscripts
1277 \IfPackageLoadedTF{realscripts}%
1278 {\RenewDocumentCommand\fakesuperscript{m}{\fup{##1}}%
1279 \NewDocumentCommand\FB@up{m}%
1280 \realsuperscript{FB@lc{##1}}%
1281 \DeclareRobustCommand*\up{}%
1282 \texorpdfstring{\@ifstar{\fup}{\FB@up}}%
1283 {}%
1284 }%
1285 }
1286 {\DeclareRobustCommand*\up{}%
1287 \texorpdfstring{\@ifstar{\fup}{\fup}}%

```

```

1288             {}%
1289         }%
1290     }
1291 \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).

```

1292     \NewDocumentCommand\FB@up{m}{%
1293         \textsuperscript{\FB@lc{##1}}}%
1294     \DeclareRobustCommand*\up{}%
1295         \texorpdfstring{\@ifstar{\fup}{\FB@up}}}%
1296         {}%
1297     }%
1298 \fi

```

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

```

1299 \ifFBLowercaseSuperscripts
1300 \else
1301     \renewcommand*\FB@lc[1]{##1}%
1302 \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.

```

1303 \ifFBUnicodeNoBreakSpaces
1304     \FB@ucsNBSP=\@ne
1305     \renewcommand*\FBmedkern{\char"202F\relax}%
1306     \renewcommand*\FBthickkern{\char"A0\relax}%
1307     \ifFBThinSpaceInFrenchNumbers
1308         \renewcommand*\FBthousandsep{\char"202F\relax}%
1309     \else
1310         \renewcommand*\FBthousandsep{\char"A0\relax}%
1311     \fi
1312 \fi

```

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

```

1313 \ifFB@koma
1314     \ifFBTocPartNameFull
1315         \KOMAOPTIONS{toc=flat, numbers=nodotatend}%
1316         \renewcommand*\addparttocentry[2]{%
1317             \addtocentrydefault{part}{\FBtocpartname{##1}}{##2}}%
1318     \fi
1319 \fi
1320 \@ifclassloaded{memoir}%
1321     {\ifFBTocPartNameFull

```

```

1322     \renewcommand{\partnumberline}[1]{\FBtocpartname{##1}}%
1323     \fi
1324 }{}%

```

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

```

1325 \ifFBShowOptions
1326   \GenericWarning{* }{%
1327     *** List of possible options for babel-french ***\MessageBreak
1328     [Default values between brackets when french is loaded *LAST*]%
1329     \MessageBreak
1330     ShowOptions [false]\MessageBreak
1331     StandardLayout [false]\MessageBreak
1332     PartNameFull [true]\MessageBreak
1333     TocPartNameFull [true]\MessageBreak
1334     IndentFirst [true]\MessageBreak
1335     ListItemsAsPar [false]\MessageBreak
1336     StandardListSpacing [false]\MessageBreak
1337     StandardItemizeEnv [false]\MessageBreak
1338     StandardEnumerateEnv [false]\MessageBreak
1339     StandardItemLabels [false]\MessageBreak
1340     ItemLabels=\textemdash, \textbullet,
1341       \protect\ding{43},... [\textendash]\MessageBreak
1342     ItemLabeli=\textemdash, \textbullet,
1343       \protect\ding{43},... [\textendash]\MessageBreak
1344     ItemLabelii=\textemdash, \textbullet,
1345       \protect\ding{43},... [\textendash]\MessageBreak
1346     ItemLabeliii=\textemdash, \textbullet,
1347       \protect\ding{43},... [\textendash]\MessageBreak
1348     ItemLabeliv=\textemdash, \textbullet,
1349       \protect\ding{43},... [\textendash]\MessageBreak
1350     StandardLists [false]\MessageBreak
1351     FrenchFootnotes [true]\MessageBreak
1352     AutoSpaceFootnotes [true]\MessageBreak
1353     AutoSpacePunctuation [true]\MessageBreak
1354     ThinColonSpace [false]\MessageBreak
1355     ThinSpaceInFrenchNumbers [false]\MessageBreak
1356     UnicodeNoBreakSpaces [false]\MessageBreak
1357     OriginalTypewriter [false]\MessageBreak
1358     INGuillSpace [false]\MessageBreak
1359     EveryParGuill=open, close, none [open]\MessageBreak
1360     EveryLineGuill=open, close, none
1361       [open in LuaTeX, none otherwise]\MessageBreak
1362     InnerGuillSingle [false]\MessageBreak
1363     SmallCapsFigTabCaptions [true]\MessageBreak
1364     FrenchSuperscripts [true]\MessageBreak
1365     LowercaseSuperscripts [true]\MessageBreak
1366     SuppressWarning [false]\MessageBreak

```

```

1367 \MessageBreak
1368 *****%
1369 \MessageBreak\protect\ frenchsetup{ShowOptions}}
1370 \fi
1371 }

```

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. Force `\FBprocess@options` to be processed only once.

```

1372 \AddToHook{env/document/before}{%
1373 \providecommand*\xspace{\relax}%
1374 \FBprocess@options
1375 \renewcommand*\FBprocess@options{\relax}%
1376 }

```

2.10 French lists

`\listFB` 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.

```

1377 \let\listORI\list
1378 \let\endlistORI\endlist
1379 \newdimen\FB@pardim
1380 \def\FB@listVsettings{%
1381 \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
1382 \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
1383 \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
1384 \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`.

```

1385 \FB@pardim=\parskip

```

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

```

1386 \ifdim\FB@pardim>\z@
1387 \addtolength{\topsep}{-\FB@pardim}%
1388 \addtolength{\partopsep}{\FB@pardim}%
1389 \setlength{\parsep}{\FB@pardim}%
1390 \addtolength{\itemsep}{-\FB@pardim}%
1391 \fi
1392 }
1393 \def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
1394 \let\endlistFB\endlistORI

```

Let's now 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;
- In French the labels of itemize-lists are vertically aligned as shown p. 6.

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

```

\Frlabelitemii 1395 \renewcommand*{\FrenchLabelItem}{\textemdash}
\Frlabelitemiii 1396 \renewcommand*{\Frlabelitemi}{\FrenchLabelItem}
\Frlabelitemiv 1397 \renewcommand*{\Frlabelitemii}{\FrenchLabelItem}
1398 \renewcommand*{\Frlabelitemiii}{\FrenchLabelItem}
1399 \renewcommand*{\Frlabelitemiv}{\FrenchLabelItem}

```

\listindentFB Let's define four dimens `\listindentFB`, `\descindentFB`, `\labelindentFB` and

\descindentFB `\labelwidthFB` to customise lists' horizontal indentations. They are given silly neg-

\labelindentFB ative values here in order to eventually enable their customisation in the preamble.

\labelwidthFB They will get reasonable defaults later when entering French (see below `\setlistindentFB` and `\setlabelitemsFB`) unless they have been customised before.

```

1400 \newdimen\listindentFB
1401 \setlength{\listindentFB}{-1pt}
1402 \newdimen\descindentFB
1403 \setlength{\descindentFB}{-1pt}
1404 \newdimen\labelindentFB
1405 \setlength{\labelindentFB}{-1pt}

```

```

1406 \newdimen\labelwidthFB
1407 \setlength{\labelwidthFB}{-1pt}

```

The next function will be included in \update@frenchlists which is executed in \extrasfrench{} ‘AtBeginDocument’.

```

1408 \def\setlistindentFB{%
1409   \ifdim\labelindentFB<\z@
1410     \ifdim\parindent=\z@
1411       \setlength{\labelindentFB}{1.5em}%
1412     \else
1413       \setlength{\labelindentFB}{\parindent}%
1414     \fi
1415   \fi
1416   \ifdim\listindentFB<\z@
1417     \ifdim\parindent=\z@
1418       \setlength{\listindentFB}{1.5em}%
1419     \else
1420       \setlength{\listindentFB}{\parindent}%
1421     \fi
1422   \fi
1423   \ifdim\descindentFB<\z@
1424     \ifFBListItemsAsPar
1425       \setlength{\descindentFB}{\labelindentFB}%
1426     \else
1427       \setlength{\descindentFB}{\listindentFB}%
1428     \fi
1429   \fi
1430 }

```

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

```

1431 \newdimen\leftmarginFB
1432 \def\FB@listHsettings{%
1433   \ifFBListItemsAsPar

```

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

```

1434     \itemindent=\labelindentFB
1435     \advance\itemindent by \labelwidthFB
1436     \advance\itemindent by \labelsep
1437     \leftmargini\z@
1438     \bbl@for\FB@dp {2, 3, 4, 5, 6}%
1439       {\csname leftmargin\romannumeral\FB@dp\endcsname =
1440         \labelindentFB}%
1441   \else

```

Default layout: labels hanging into the list left margin.

```

1442     \leftmarginFB=\labelwidthFB

```

```

1443 \advance\leftmarginFB by \labelsep
1444 \bbl@for\FB@dp {1, 2, 3, 4, 5, 6}%
1445   {\csname leftmargin\romannumeral\FB@dp\endcsname =
1446     \leftmarginFB}%
1447 \advance\leftmargini by \listindentFB

```

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

```

1448 \listparindent=\parindent
1449 \fi
1450 \leftmargin=\csname leftmargin%
1451   \ifnum\@listdepth=\@ne i\else ii\fi\endcsname
1452 }

```

\itemizeFB New environment for French itemize-lists.

\FB@itemizesettings \FB@itemizesettings does two things: first suppress all vertical spaces including glue unless option **StandardListSpacing** is set, then set horizontal indentations according to \FB@listHsettings.

```

1453 \def\FB@itemizesettings{%
1454   \ifFBStandardListSpacing
1455   \else
1456     \FB@pardim=\parskip
1457     \ifdim\FB@pardim>\z@
1458       \setlength{\topsep}{-\FB@pardim}%
1459       \setlength{\partopsep}{\FB@pardim}%
1460       \setlength{\parsep}{\FB@pardim}%
1461       \setlength{\itemsep}{-\FB@pardim}%
1462     \else
1463       \setlength{\topsep}{\z@}%
1464       \setlength{\partopsep}{\z@}%
1465       \setlength{\parsep}{\z@}%
1466       \setlength{\itemsep}{\z@}%
1467     \fi
1468   \fi
1469   \settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
1470   \FB@listHsettings
1471 }

```

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

```

1472 \def\itemizeFB{%
1473   \ifnum \@itemdepth >\thr@@\@toodeep\else
1474     \advance\@itemdepth by \@ne
1475     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
1476     \expandafter
1477     \listORI
1478     \csname\@itemitem\endcsname
1479     \FB@itemizesettings

```



```

1480   \fi
1481 }
1482 \let\enditemizeFB\endlistORI

```

The next function will be included in `\update@frenchlists` which is executed in `\extrasfrench{}` ‘AtBeginDocument’.

```

1483 \def\setlabelitemsFB{%
1484   \let\labelitemi\Frlabelitemi
1485   \let\labelitemii\Frlabelitemii
1486   \let\labelitemiii\Frlabelitemiii
1487   \let\labelitemiv\Frlabelitemiv
1488   \ifdim\labelwidthFB<\z@
1489     \settowidth{\labelwidthFB}{\FrenchLabelItem}%
1490   \fi
1491 }

```

\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`.

```

1492 \def\enumerateFB{%
1493   \ifnum \@enumdepth >\thr@@\toodeep\else
1494     \advance\@enumdepth by \@ne
1495     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
1496     \expandafter
1497     \list
1498       \csname label\@enumctr\endcsname
1499       {\FB@listHsettings
1500        \usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
1501   \fi
1502 }
1503 \let\endenumerateFB\endlistORI

```

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

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.

```

1504 \def\descriptionFB{%
1505   \list{}{\FB@listHsettings
1506     \labelwidth=\z@
1507     \ifFBListItemsAsPar
1508       \itemindent=\descindentFB

```

```

1509         \else
1510             \itemindent=-\leftmargin
1511             \ifnum\@listdepth=\@ne
1512                 \ifdim\descindentFB=\z@
1513                     \ifdim\listindentFB>\z@
1514                         \leftmargini=\listindentFB
1515                         \leftmargin=\leftmargini
1516                         \itemindent=-\leftmargin
1517                     \fi
1518                 \else
1519                     \advance\itemindent by \descindentFB
1520                 \fi
1521             \fi
1522         \fi
1523         \let\makelabel\descriptionlabel}%
1524 }
1525 \let\enddescriptionFB\endlistORI

```

\bbl@frenchlistlayout **\update@legacylists** will set up lists according to the final options (default or part of **\frenchsetup{}** eventually overruled in **\FBprocess@options**). This is for conventional lists *only*.

```

1526 \def\update@legacylists{%
1527   \setlistindentFB
1528   \ifFBStandardListSpacing
1529   \else \let\list\listFB \fi
1530   \ifFBStandardItemizeEnv
1531   \else \let\itemize\itemizeFB \fi
1532   \ifFBStandardItemLabels
1533   \else \setlabelitemsFB \fi
1534   \ifFBStandardEnumerateEnv
1535   \else \let\enumerate\enumerateFB \let\description\descriptionFB \fi
1536 }

```

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 updated once for all. There is nothing to do for lists in **\noextrasfrench**.

Lists' layout no longer changes at language switches.

```

1537 \def\bbl@frenchlistlayout{%
1538   \ifFB@mainlanguage@FR
1539   \ifFBnewlists
1540   \else
1541     \update@legacylists
1542     \let\update@legacylists\relax
1543   \fi
1544   \fi}
1545 \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 in only if `IndentFirst=true` and French isn't the main language.

```
1546 \def\bbl@frenchindent{%
1547   \ifFBIndentFirst
1548     \ifFB@mainlanguage@FR\else\babel@save\@afterindentfalse\fi
1549     \let\@afterindentfalse\@afterindenttrue
1550     \@afterindenttrue
1551   \fi}
1552 \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 *does not depend* on the current language (just think of two footnotes on the same page looking different because one was called in a French part, the other one in English!).

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

\parindentFFN The value of `\parindentFFN` will be redefined at the `\begin{document}`, as the maximum of `\parindent` and 1.8em *unless* it has been set in the preamble (the weird value `\dotFFN` 10in is just for testing whether `\parindentFFN` has been set or not).

```
1553 \newdimen\parindentFFN
1554 \parindentFFN=\maxdimen
```

`\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.

```
1555 \newdimen\FBfnindent
1556 \newcommand*{\dotFFN}{.}
1557 \newcommand*{\kernFFN}{\kern .5em}
```

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

```
1558 \newcommand*{\FBfnmarkspace}{\kern .5\fontdimen2\font}
```

\newfootnotemarkFB This code is for the new footnotes templates.

```
1559 \newcommand*{\newfootnotemarkFB}{%
1560   \setbox\@tempboxa\hbox{\@thefnmark}%
1561   \ifdim\wd\@tempboxa>\z@
1562     \llap{\@thefnmark}\dotFFN\kernFFN
1563   \fi}
```

This code is 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.

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.

```
1564 \iffB@koma
1565 \let\@makefntextORI\@makefntext
1566 \let\@@makefnmarkORI\@@makefnmark
```

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

```
1567 \deffootnote[\FBfnindent]{\z@}{\parindentFFN}%
1568           {\thefootnotemark\dotFFN\kernFFN}
1569 \let\@makefntextFB\@makefntext
1570 \let\@@makefnmarkFB\@@makefnmark
```

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

```
1571 \deffootnote[\parindentFFN]{\z@}{\parindentFFN}%
1572           {\textsuperscript{\thefootnotemark}}
1573 \let\@makefntextTH\@makefntext
1574 \let\@@makefnmarkTH\@@makefnmark
```

Restore the original definitions.

```
1575 \let\@makefntext\@makefntextORI
1576 \let\@@makefnmark\@@makefnmarkORI
1577 \fi
```

Definitions for the `memoir` class:

```
1578 \ifclassloaded{memoir}
```

(see original definition in `memman.pdf`)

```
1579 {\newcommand{\@makefntextFB}[1]{%
1580   \def\footscript##1{##1\dotFFN\kernFFN}%
1581   \setlength{\footmarkwidth}{\FBfnindent}%
1582   \setlength{\footmarksep}{-\footmarkwidth}%
1583   \setlength{\footparindent}{\parindentFFN}%
1584   \makefootmark #1}%
1585 }}
```

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.

```

1586 \ifclassloaded{beamer}
1587   {\def\@makefntextFB#1{%
1588     \def\insertfootnotetext{#1}%
1589     \def\insertfootnotemark{\insertfootnotemarkFB}%
1590     \usebeamertemplate***{footnote}}}%
1591   \def\insertfootnotemarkFB{%
1592     \usebeamercolor[fg]{footnote mark}%
1593     \usebeamerfont*{footnote mark}%
1594     \llap{\@thefnmark}\dotFFN\kernFFN}%
1595   }{}

```

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)!

```

1596 \providecommand*\insertfootnotemarkFB{%
1597   \parindent=\parindentFFN
1598   \rule\z@\footnotesep
1599   \setbox\@tempboxa\hbox{\@thefnmark}%
1600   \ifdim\wd\@tempboxa>\z@
1601     \llap{\@thefnmark}\dotFFN\kernFFN
1602   \fi}
1603 \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`.

```

1604 \def\@footnotemarkFB{\leavevmode\unskip\unkern
1605   \protect\FBfnmarkspace\@footnotemarkORI}%

```

The following command `\FBlegacyfootnote@switch` gathers the code needed to switch between French or Standard layout for footnotes; it is processed in `\FBprocess@options` just before `\begin{document}`.

The LuaTeX command `\localleftbox` and `\FBeverypar@quote` used by `\frquote{}` have to be reset inside footnotes; done for LaTeX based formats only.

```
1606 \newcommand*{\FBlegacyfootnote@switch}{%
```

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

```
1607 \IfPackageLoadedTF{footnotebackref}%
1608   {\FBFrenchFootnotesfalse
1609     \PackageWarning{french.ldf}%
1610       {footnotebackref package loaded.\MessageBreak
1611         babel-french will NOT customise footnotes;%
1612         \MessageBreak reported}}%
1613   {}%
```

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.

```
1614 \IfPackageLoadedTF{bigfoot}%
1615   {\PackageWarning{french.ldf}%
1616     {bigfoot package in use.\MessageBreak
1617       babel-french will NOT customise footnotes;%
1618       \MessageBreak reported}}%
```

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

```
1619   {\let\@footnotemarkORI\@footnotemark
1620     \ifFBAutoSpaceFootnotes
1621       \let\@footnotemark\@footnotemarkFB
1622     \fi
1623     \ifdim\parindentFFN<\maxdimen
1624     \else
1625       \parindentFFN=\parindent
1626       \ifdim\parindentFFN<1.5em \parindentFFN=1.5em \fi
1627     \fi
1628     \settowidth{\FBfnindent}{\dotFFN\kernFFN}%
1629     \addtolength{\FBfnindent}{\parindentFFN}%
1630     \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 mandatory argument of \deffootnote{} (used as \leftskip) is non-nil (default is 1em, 0pt in French).

```
1631   \ifFB@koma
1632     \let\@@makefnmarkORI\@@makefnmark
1633     \long\def\@makefntext##1{%
1634       \localleftbox{}%
1635       \let\FBeverypar@save\FBeverypar@quote
1636       \let\FBeverypar@quote\relax
1637       \ifFBFrenchFootnotes
1638         \ifx\footnote\thanks
```

```

1639         \let\@@makefnmark\@@makefnmarkTH
1640         \@makefnmarkTH{##1}
1641     \else
1642         \let\@@makefnmark\@@makefnmarkFB
1643         \@makefnmarkFB{##1}
1644     \fi
1645 \else
1646     \let\@@makefnmark\@@makefnmarkORI
1647     \@makefnmarkORI{##1}%
1648 \fi
1649 \let\FBeverypar@quote\FBeverypar@save
1650 \localleftbox{\FBeveryline@quote}}}%
1651 \else

```

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

```

1652     \ifclassloaded{memoir}%
1653     {
1654         \setlength{\thanksmarkwidth}{\parindentFFN}%
1655         \setlength{\thanksmarksep}{-\thanksmarkwidth}%
1656     }
1657 \fi

```

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

```

1658     \ifclassloaded{beamer}%
1659     {
1660         \ifFBFrenchFootnotes
1661         \ifdim\parindentFFN=1.5em\else
1662             \FBWarning{%
1663                 \protect\parindentFFN\space is ineffective%
1664                 \MessageBreak within the beamer class.%
1665                 \MessageBreak Reported}%
1666         \fi
1667     }

```

Definition of \@makefnmark for all other classes:

```

1668     \long\def\@makefnmark##1{%
1669         \localleftbox{}%
1670         \let\FBeverypar@save\FBeverypar@quote
1671         \let\FBeverypar@quote\relax
1672         \ifFBFrenchFootnotes
1673             \@makefnmarkFB{##1}%
1674         \else
1675             \@makefnmarkORI{##1}%
1676         \fi
1677         \let\FBeverypar@quote\FBeverypar@save
1678         \localleftbox{\FBeveryline@quote}}}%

```

```
1679      \fi
```

Revert to standard footnotes inside minipages.

```
1680      \AddToHook{env/minipage/begin}{\FBFrenchFootnotesfalse}
```

```
1681    }%
```

```
1682  }
```

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

```
1683 \def\bbl@frenchfootnotes{%
```

```
1684   \ifFB@mainlanguage@FR
```

```
1685     \ifFBnewfootnotes
```

```
1686     \else
```

```
1687       \FBlegacyfootnote@switch
```

```
1688       \let\FBlegacyfootnote@switch\relax
```

```
1689     \fi
```

```
1690   \fi}
```

```
1691 \addto\extrasfrench{\bbl@frenchfootnotes}
```

2.13 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.

```
1692 \FBclean@on@exit
```

```
1693 \ldf@finish\CurrentOption
```

```
1694 \let\loadlocalcfg\FB@llc
```

```
1695 </french>
```


3 Change History

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

v4.0d	General: \FBprocess@options must be processed only once, reported by Herbert Voß for dtk.cls.	53	and print a warning.	64
v4.0c	General: \StandardFootnotes and \FrenchFootnotes commands removed.	64	v3.6b	\frquote: Flag \ifFBcloseguill does not apply to \@fgii.
v4.0b	General: \FBlegacyfootnote@switch moved to \extrasfrench (tagging issue).	64	v3.6a	\NoAutoSpacing: \NoAutoSpacing must be inhibited in bookmarks.
	New \ifFBnewlists and \ifFBnewfootnotes to handle the new corresponding templates.	40		\@footnotemark: Allow customisation of the space added in \@footnotemarkFB.
	\FBprocess@options: New code to customise footnotes when the new templates are available.	48	v3.5s	frenchb.lua: A ‘:’ followed by ‘-’ or a ligature should not trigger spacing.
v4.0a	General: New customisation for the Part entries in the toc. Suggested by Julien Labbé.	39	v3.5q	\listFB: Bug correction: \parsep should be related to \parskip and \listparindent to \parindent.
	Option CustomiseFigTabCaptions is set to false.	40	v3.5p	\DecimalMathComma: \DecimalMathComma can again be used in the preamble for a global action. It now works as expected inside a group.
	Options og and fg are now useless.	40	v3.5o	frenchb.lua: Opening guill.: look ahead when next is a penalty (nobreak space).
	Removed obsolete compatibility options GlobalLayoutFrench, ListOldLayout, OldFigTabCaption	40	v3.5k	\bsc: \bsc now relies on \texorpdfstring to be safe in bookmarks.
	frenchb.lua: Codes 0x2039 and 0x203A added for French single quotes.	18	v3.5h	frenchb.lua: Added glues and penalties should inherit attributes from the related punctuation character; this is mandatory for Lua-UL to underline and highlight them. Thanks to Marcel Krüger for providing the fix.
	New function ‘euphonic_t’ for hyphenation of compound words. Suggested by Thomas Savary.	26	v3.5g	frenchb.lua: The kerning callback is a bit specific: adding code with
v3.7a	General: Support for acadian dropped. The files acadian.ldf, canadien.ldf, frenchb.ldf and francais.ldf load french.ldf			

add_to_callback actually deletes the legacy kerning as pointed out by Marcel Krüger on SE.	21	hanging into the left margin. . . .	55
v3.5c		v3.4a	
General: Remove grouping inside <code>\@makefn</code> text, <code>\localleftbox</code> and <code>\FBeverypar@quote</code> saved and restored instead.	61	General: Shrink/stretch removed in <code>\FBthousandsep</code>	38
v3.5b		v3.3d	
General: Reset <code>\FBeverypar@quote</code> locally inside <code>\@makefn</code> text. Needed by <code>\frquote</code>	61	<code>frenchb.lua</code> : In default mode, for ‘:’ only, check if next node is a glyph or not. If it is, turn the ‘auto’ flag to false (avoids spurious spaces in URLs, MSDOS paths or 10:35). . . .	22
v3.5a		v3.3c	
General: New optional layout for lists: lists’ items can be typeset as paragraphs with indented labels while the default leaves the labels		General: New command <code>\FBthousandsep</code> to customise numprint.	38
		Reset <code>\localleftbox</code> locally inside <code>\@makefn</code> text. Needed by <code>\frquote</code> with LuaTeX.	61